UNIVERSITY OF CALIFORNIA

Los Angeles

Stochastic Optimization and Subgraph Search

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Jacob Moorman

2021

ABSTRACT OF THE DISSERTATION

Stochastic Optimization and Subgraph Search

by

Jacob Moorman
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2021
Professor Andrea Bertozzi, Co-Chair
Professor Deanna M. Hunter, Co-Chair

In this thesis, we study randomized algorithms for numerical linear algebra and pattern matching algorithms for multiplex networks. We first analyze the convergence of two classes of randomized iterative methods for solving large linear systems of equations. In particular, we analyze sketch-and-project methods with adaptive sampling strategies and parallelized randomized Kaczmarz methods with averaging. We observe empirically that the convergence of these methods reflects the worst-case convergence theory. We later discuss subgraph matching and various related problems including inexact search. We introduce filtering algorithms that are specialized to multiplex networks. In both the exact and inexact settings, we aim to understand the entire solution space rather than simply finding one match for a given pattern. We observe that there are often a combinatorially large number of matches depending on the amount of symmetry in the pattern and data.

The dissertation of Jacob Moorman is approved.

Luminita Aura Vese

Mason Alexander Porter

Deanna M. Hunter, Committee Co-Chair

Andrea Bertozzi, Committee Co-Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisors Prof. Deanna Needell (Deanna M. Hunter) and Prof. Andrea Bertozzi for their supervision and support during my studies. I would also like to thank my collaborators for their hard work and enthusiasm. I am thankful to the members of Deanna Needell's research group for creating such an inspiring and supportive environment.

Chapter 2 is a version of [GMM19] and is joint work with Prof. Robert Gower, Denali Molitor, and Prof. Deanna Needell. Robert Gower proposed the project, and co-supervised the project with Deanna Needell. Robert Gower, Denali Molitor, and I contributed the selection rules and convergence analysis. Denali Molitor and I contributed the experiments and the computational complexity analysis.

Chapter 3 is a version of [MTM21] and is joint work with Thomas Tu, Denali Molitor, and Prof. Deanna Needell. Deanna Needell supervised the project. Thomas Tu, Denali Molitor, and I contributed the convergence analysis and the experiments.

Chapter 4 is a version of [MTC21] and is joint work with Thomas Tu, Qinyi Chen, Xie He, Denali Molitor, and Prof. Andrea Bertozzi. Prof. Andrea Bertozzi supervised the project. I developed the algorithms with help from Thomas Tu and Qinyi Chen. Thomas Tu contributed the Sudoku and multiplex Erdős–Rényi experiments. Denali Molitor contributed the crossword code and experiments. Qinyi Chen and Xie He contributed the real-world experiments. Thomas Tu, Qinyi Chen, Xie He, and I contributed the adversarial activity experiments. The primary difference between Chapter 4 and [MTC21] is the addition of Subsection 4.4.3 where I have added a description of how Denali Molitor and I were able to use a subgraph isomorphism solver to fill a crossword grid with answers. We thank Jamie Atlas, Omri Azencot, Zachary Boyd, Jeremy Budd, Yonatan Dukler, Matthew Jacobs, Blaine Keetch, Hao Li, Kevin Miller, Mason A. Porter, Bao Wang, Yotam Yaniv, and Baichuan Yuan for helpful discussions. We thank the anonymous reviewers for their valuable suggestions.

Chapter 5 is a version of [TMY20] and is joint work with Thomas Tu, Dominic Yang,

Qinyi Chen, and Prof. Andrea Bertozzi. Prof. Andrea Bertozzi supervised the project. I developed the algorithms with help from Thomas Tu and Qinyi Chen. Thomas Tu and Dominic Yang contributed the experiments. The primary difference between Chapter 5 and [TMY20] is the improvement of Subsection 5.2.3 where I added an explanation of how one of the key subroutines can be implemented efficiently. This efficient implementation was developed by Qinyi Chen and I. We thank George Chin, Joseph Cottam, Natalie Heller, Patrick Mackey, Sumit Purohit, and the rest of the team at PNNL for providing us with the AIDA dataset and the graph edit distance metric. We additionally thank Yurun Ge and Xie He for helpful comments and discussions.

I am grateful to the UCLA Graduate Division (dissertation year fellowship), the Defense Advanced Research Projects Agency (agreement number FA8750-18-2-0066), and the National Science Foundation (NSF DGE-1829071) for financial support during my PhD.

The material in Chapters 4 and 5 is based on research sponsored by the Air Force Research Laboratory and DARPA under agreement number FA8750-18-2-0066. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government.

VITA

2012–2013 Game Development Consultant, Mission Critical Studios.

2015 Software Engineering Intern, Trillium Labs LLC.

2016 B.S. (Mathematics and Computer Science), NJIT.

2017 Artificial Intelligence Research Intern, NovaSignal.

2016–2018 Teaching Assistant, Mathematics Department, UCLA.

2018 M.A. (Mathematics), UCLA.

2019 Computer Vision Research Intern, HRL Laboratories LLC.

2018–Present Graduate Student Researcher, Mathematics Department, UCLA.

PUBLICATIONS

R. M. Gower, D. Molitor, J. D. Moorman, D. Needell, "Adaptive Sketch-and-Project Methods for Solving Linear Systems." *SIAM Journal on Matrix Analysis and Applications*, 2021. To Appear. `https://arxiv.org/abs/1909.03604`.

J. D. Moorman, Q. Chen, T. K. Tu, X. He, A. L. Bertozzi, "Subgraph Matching on Multiplex Networks." *IEEE Transactions on Network Science and Engineering*, 2021. Early Access.

J. D. Moorman, T. K. Tu, D. Molitor, D. Needell, "Randomized Kaczmarz with Averaging." *BIT Numerical Mathematics*, **61**(1):337–359, 2021.

T. K. Tu, J. D. Moorman, D. Yang, Q. Chen, A. L. Bertozzi, "Inexact Attributed Subgraph Matching." In *2020 IEEE International Conference on Big Data*, pp. 2575–2582. IEEE, 2020.

J. D. Moorman, T. K. Tu, D. Molitor, D. Needell, "Randomized Kaczmarz with Averaging." In *Information Theory and Applications Workshop*. 2019.

J. D. Moorman, Q. Chen, T. K. Tu, Z. M. Boyd, A. L. Bertozzi, "Filtering Methods for Subgraph Matching on Multiplex Networks." In *2018 IEEE International Conference on Big Data*, pp. 3980-3985. IEEE, 2018.

CHAPTER 1

Introduction

With the increasing application of generic algorithms in high-impact applications such as medicine, surveillance, finance, and epidemiology, a thorough understanding of each algorithm is key. A lack of such an understanding, can lead to unexpected and significant negative consequences. In applications where data has structure that does not perfectly match a generic algorithm, an understanding of the algorithm can enable scientists to tailor it to their application. In this thesis, we focus on deepening our understanding of iterative methods for solving linear systems of equations and on specializing algorithms for pattern matching on graph-structured data to applications where the graphs have attributes on the nodes and edges.

Stochastic gradient descent (SGD) is an iterative optimization algorithm that approximates the gradient descent algorithm by replacing the full gradient of an objective function $F(x)$ by a stochastic estimate thereof [BCN18]. It is typically used to minimize objective functions of the form $F(x) = \sum_{i=1}^{n} f_i(x)$, where each $i = 1, \ldots, n$ corresponds to a datapoint d^i. Such objective functions arise in domains such as physics, engineering, statistics, imaging, and machine learning. SGD is particularly useful when the data $\{d^1, \ldots, d^n\}$ does not fit in memory since the full gradient $\nabla F(x)$ is prohibitively expensive to compute in that case.

SGD begins with an initial guess for the optimum, x^0. At each iteration k, one or more of the datapoints d^i are chosen and the guess x^k is updated based on the stochastic gradient $\nabla f_i(x^k)$ in an effort to decrease $F(x)$. When more than one datapoint is chosen, an average of the stochastic gradients is used. While convergence of SGD is well understood theoretically in restricted settings such as smooth and strongly-convex F [BCN18], it is frequently applied

successfully to problems which do not satisfy such restrictions [LBH15]. In Chapters 2 and 3, we focus on objective functions corresponding to solving linear systems of equations, where SGD is equivalent to the sketch-and-project method (SaP) [RT20] or the randomized Kaczmarz method (RK) [NSW16] respectively.

When applying SGD, there are several hyperparameters that can be tuned to improve the convergence of the algorithm. For example, one must choose the size of the step to take at each iteration. We focus on two such hyperparameters in this work: how many stochastic gradients $\nabla f_i(x^k)$ to use and how to select the indices i at each iteration. We investigate the convergence of SaP as it relates to the strategy of choosing i in Chapter 2. We study the convergence of RK as it relates to the strategy of choosing i and the number of stochastic gradients sampled in Chapter 3.

In search applications where the data has graph structure and the query can be phrased as a smaller pattern graph, finding a subgraph of the data that matches the pattern is called the *subgraph isomorphism problem (SIP)* [Ull76]. Finding all such subgraphs is called the *subgraph matching problem (SMP)* or simply *subgraph matching (SM)* [SWW12]. In Chapter 4, we discuss these and other related problems, and observe how the difficulty of the problems vary from dataset to dataset. We introduce algorithms for solving these search problems that are tailored to settings where the data and pattern graphs are endowed with labels on the nodes and edges. In Chapter 5, we introduce analogous algorithms for finding results which only approximately match the pattern.

CHAPTER 2

Adaptive Sketch-and-Project Methods*

2.1 Introduction

We consider the fundamental problem of finding an approximate solution to the linear system

$$\mathbf{A}x = b, \tag{2.1}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Here, we consider consistent systems, for which there exists an x that satisfies Equation (2.1). We make no assumptions on the shape or rank of $\mathbf{A}$. If $\mathbf{A}$ has full rank with as many columns as rows (i.e., $n \geq m$), the system is always consistent. Otherwise, only $b \in \text{Range}(\mathbf{A})$ will yield a consistent system. Given the possibility of multiple solutions, we set out to find a least-norm solution given by

$$x^\star \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \tfrac{1}{2}\|x\|_{\mathbf{B}}^2 \quad \text{subject to} \quad \mathbf{A}x = b, \tag{2.2}$$

where $\mathbf{B} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $\|x\|_{\mathbf{B}}^2 \overset{\text{def}}{=} \langle \mathbf{B}x, x\rangle$.

In modern contexts, matrix vector projects $\mathbf{A}x$ can be prohibitively expensive due to the size of $\mathbf{A}$. In such contexts, direct methods for solving Equation (2.2) can be infeasible, and iterative methods are favored. In particular, Krylov subspace iterative methods including the conjugate gradient algorithms [HS52] are the industrial standard so long as one can afford full matrix vector products [GV13]. However, when $\mathbf{A}$ is too large to fit in memory, then randomized iterative methods such as randomized Kaczmarz [Kac37, SV09] and coordinate descent [MNR15, LL10] can be used since they require only part of the matrix per iteration.

*This chapter is adapted from [GMM19].

2.1.1 Randomized Kacmarz

The randomized Kaczmarz method is typically used to solve linear systems of equations in the large data regime, when the number of samples m is much larger than the dimension n. The Kaczmarz method was originally proposed in 1937 and has seen applications in computer tomography (CT scans), signal processing, and other areas [Kac37, SV09, GBH70, Nat01]. In each iteration k, the current iterate x^k is projected onto the solution space of a selected row of the linear system of Equation (2.1). Specifically, at each iteration,

$$x^{k+1} = \underset{x \in \mathbb{R}^n}{\arg\min} \|x - x^k\|^2 \quad \text{subject to} \quad \mathbf{A}_{i_k:} x = b_{i_k},$$

where $\mathbf{A}_{i_k:}$ is the row of $\mathbf{A}$ selected at iteration k. Let $\mathbf{A}_{i_k:}^\top$ denote the transpose of this row. The Kaczmarz update can be written explicitly as

$$x^{k+1} = x^k + \frac{b_{i_k} - \langle \mathbf{A}_{i_k:}, x^k \rangle}{\|\mathbf{A}_{i_k:}\|^2} \mathbf{A}_{i_k:}^\top. \tag{2.3}$$

Randomized Kaczmarz can also be interpreted as a special case of stochastic gradient descent (SGD) applied to the loss function [NSW16]

$$F(x) = \sum_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} \frac{1}{2} (\mathbf{A}_{i:} x - b_i)^2.$$

2.1.2 Coordinate descent

Coordinate descent is commonly used for optimizing general convex optimization functions when the dimensions are extremely large, since at each iteration only a single coordinate (or dimension) is updated [RT14, RT16]. Here, we consider coordinate descent applied to Equation (2.2). In this setting, it is sometimes referred to as randomized Gauss–Seidel [MNR15, LL10].

At iteration k an index $i \in \{1, \ldots, n\}$ is selected and the coordinate x_i^k of the current iterate x^k is updated such that the least-squares objective $\|b - \mathbf{A}x\|^2$ is minimized. More formally,

$$x^{k+1} = \underset{x \in \mathbb{R}^n,\, \lambda \in \mathbb{R}}{\arg\min} \|b - \mathbf{A}x\|^2 \quad \text{subject to} \quad x = x^k + \lambda\, e^i,$$

where e^i is the i^{th} coordinate vector. Let $\mathbf{A}_{:i}$ denote the i^{th} column of $\mathbf{A}$ and $\mathbf{A}_{:i}^\top$ denote the transpose of this column. The explicit update for coordinate descent applied to Equation (2.2) is given by

$$x^{k+1} = x^k - \frac{\mathbf{A}_{:i_k}^\top(\mathbf{A}x^k - b)}{\|\mathbf{A}_{:i_k}\|^2} e^{i_k}. \tag{2.4}$$

2.1.3 Sketch-and-project methods

Sketch-and-project is a general archetypal method that unifies a variety of randomized iterative methods including both randomized Kaczmarz and coordinate descent along with their block variants [GR15b]. At each iteration, sketch-and-project methods project the current iterate onto a subsampled or sketched linear system with respect to some norm. Let $\mathbf{B} \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. We will consider the projection with respect to the $\mathbf{B}$–norm given by $\|\cdot\|_{\mathbf{B}} \overset{\text{def}}{=} \sqrt{\langle \cdot, \mathbf{B}\cdot\rangle}$.

Let $\mathbf{S}_i \in \mathbb{R}^{m\times\tau}$ for $i = 1, \ldots, q$ be the *sketching matrices* where $\tau \in \mathbb{N}$ is the *sketch size*. In general, the set of sketching matrices $\mathbf{S}_i$ could be infinite, however, here, we restrict ourselves to a finite set of $q \in \mathbb{N}$ sketching matrices. At the k^{th} iteration of the sketch-and-project method, a sketching matrix $\mathbf{S}_i$ is selected and the current iterate x^k is projected onto the solution space of the sketched system $\mathbf{S}_{i_k}^\top \mathbf{A}x = \mathbf{S}_{i_k}^\top b$ with respect to the $\mathbf{B}$–norm $\|\cdot\|_{\mathbf{B}} \overset{\text{def}}{=} \sqrt{\langle \cdot, \mathbf{B}\cdot\rangle}$. Mathematically, given a selected index $i_k \in \{1, \ldots, q\}$ the sketch-and-project update is

$$x^{k+1} = \underset{x\in\mathbb{R}^n}{\arg\min} \|x - x^k\|_{\mathbf{B}}^2 \quad \text{subject to} \quad \mathbf{S}_{i_k}^\top \mathbf{A}x = \mathbf{S}_{i_k}^\top b. \tag{2.5}$$

The closed form solution to Equation (2.5) is given by

$$x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b), \tag{2.6}$$

where

$$\mathbf{H}_i \overset{\text{def}}{=} \mathbf{S}_i(\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger \mathbf{S}_i^\top, \quad \text{for } i = 1, \ldots, q, \tag{2.7}$$

and $\dagger$ denotes the Moore–Penrose pseudoinverse.

One can recover the randomized Kaczmarz method under the sketch-and-project framework

by choosing the matrix $\mathbf{B}$ as the identity matrix and sketches $\mathbf{S}_i = e^i$. If instead $\mathbf{B} = \mathbf{A}^\top\mathbf{A}$ and sketches $\mathbf{S}_i = \mathbf{A}e^i = \mathbf{A}_{:i}$, then the resulting method is coordinate descent.

2.1.4 Sampling of indices

An important component of the methods above is the selection of the index i_k at iteration k. Methods often use independently and identically distributed (i.i.d.) indices, as this choice makes the method and analysis relatively simple [SV09, Nes12]. In addition to choosing indices i.i.d. at each iteration, several adaptive sampling methods have also been proposed, which we discuss next. These sampling strategies use information about the current iterate in order to improve convergence guarantees over i.i.d. random sampling strategies at the cost of extra calculation per iteration. Under certain conditions, such strategies can be implemented with an affordable increase in cost discussed in Sections 2.A and 2.8.

2.1.4.1 Sampling for the Kaczmarz method

The original Kaczmarz method cycles through the rows of the matrix $\mathbf{A}$ and makes projections onto the solution space with respect to each row [Kac37]. In 2009, Strohmer and Vershynin proved that sampling the rows randomly with probabilities proportional to the squared row norms (i.e. $p_i \propto \|\mathbf{A}_{i:}\|_2^2$) leads to provably exponential convergence [SV09].

Several adaptive selection strategies have also been proposed in the Kaczmarz setting. The max-distance Kaczmarz or Motzkin's method selects the index i_k at iteration k that leads to the largest magnitude update [NSL16, MS54]. In addition to the max-distance selection rule, Nutini et al. also considered the greedy selection rule that chooses the row corresponding to the maximal residual component, i.e., $i_k = \arg\max_i |\mathbf{A}_{i:}x^k - b_i|$ at each iteration, but showed that the max-distance Kacmzarz method performs at least as well as this strategy [NSL16]. More complicated adaptive methods have also been suggested for randomized Kaczmarz, such as the capped sampling strategies proposed in [BW18a, BW18b, BW19a] or the sampling Kaczmarz Motzkin's method of [LHN17].

2.1.4.2 Sampling for coordinate descent

For coordinate descent, several works have investigated adaptive coordinate selection strategies [PCJ17, NSL15, Nes12, AG18]. As coordinate descent is not restricted to solving linear systems, these works often considered more general convex loss functions. A common greedy selection strategy for coordinate descent applied to differentiable loss functions is to select the coordinate that corresponds to the maximal gradient component, which is known as the Gauss–Southwell rule [Tse90, LT92, NSL15, Nes12] or adaptively according to a duality gap [CQR15].

2.1.4.3 Sampling for sketch-and-project

The problem of determining the optimal fixed probabilities with which to select the index i_k at each iteration k was shown in Section 5.1 of [GR15b] to be a convex semi-definite program, which is a harder problem than solving the original linear system. The problem of determining the optimal adaptive probabilities is even harder as one must consider the effects of the current index selection on the future iterates. Here, instead, we present adaptive sampling rules that are not necessarily optimal, but are proven to have faster convergence guarantees than certain fixed non-adaptive rules.

2.1.5 Choosing the sketches and preconditioning

Another key question is how we should choose the set of sketching matrices. This question has been partially answered in Section 5.2 of [GR17], wherein the authors show that if a preconditioned $\mathbf{A}$ were available, then the set of sketching matrices should be drawn from row partitions or column partitions of this preconditioner. This strategy can be combined with any index sampling rule for an overall faster algorithm. Here, we will assume a set of sketching matrices has been provided, and focus only on the index sampling rule.

2.1.6 Additional related works

Various related works consider extensions to solving Equation (2.2) in the randomized Kaczmarz, coordinate descent and sketch-and-project settings. The following summary of related works is not exhaustive. While we consider consistent linear systems, others have analyzed variants of the Kaczmarz and coordinate descent methods to handle inconsistent linear systems [PP16, ZF13, Pop99, MNR15, Dum15]. An adaptive maximum-residual sampling strategy has also been analyzed for the inconsistent extension [PP16]. The randomized Kaczmarz method has also been studied in the context of solving systems of linear inequalities [LL10, MS54, BN15, BW19b]. Block and accelerated variants of randomized Kaczmarz and coordinate descent have also been analyzed [RT14, Nec19, NZZ15, NT14, LX15, NNG17, NS17b]. Recent works have considered combining ideas from random sketching methods with those from the sketch-and-project framework [PJM19].

2.2 Contributions

Adaptive sampling strategies have not yet been analyzed for the general sketch-and-project framework. We introduce three different adaptive sampling rules for the general sketch-and-project method: the max-distance sampling rule, the capped-adaptive sampling rule, and proportional sampling probabilities. We prove that each of these methods converge exponentially in mean squared error with convergence guarantees that are strictly faster than the guarantees for sampling indices uniformly.

2.2.1 Key quantity: Sketched loss

As we will see in Section 2.7, the convergence at each iteration depends on the current iterate x^k and a key quantity known as the sketched loss

$$f_i(x^k) \stackrel{\text{def}}{=} \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}. \tag{2.8}$$

Here, we use the notation $\|\cdot\|_{\mathbf{H}_i} \stackrel{\text{def}}{=} \sqrt{\langle \cdot, \mathbf{H}_i \cdot \rangle}$. Recall that $\mathbf{H}_i$, defined in Equation (2.7), is symmetric positive semi-definite and thus $\|\cdot\|_{\mathbf{H}_i}$ gives a semi-norm. The sketched loss $f_i(x^k)$ corresponding to sketch $\mathbf{S}_i$ captures the amount by which an iteration using $\mathbf{S}_i$ would reduce the squared error. This sketched loss was introduced in [RT20] where the authors show that the sketch-and-project method can be seen as a stochastic gradient method (we expand on this in Section 2.4). We show that using adaptive selection rules based on the sketched losses results in new methods with faster convergence guarantees.

2.2.2 Max-distance rule

We introduce the max-distance sketch-and-project method, which is a generalization of both the max-distance Kaczmarz method (also known as Motzkin's method) [NSL16, MS54, HN19], greedy coordinate descent (Gauss-Southwell rule [NSL15]), and their block variants. Nutini et al. showed that the max-distance Kaczmarz method performs at least as well as uniform sampling and the non-uniform sampling method of [SV09], in which rows are sampled with probabilities proportional to the squared row norms of $\mathbf{A}$ [NSL16]. We extend this result to the general sketch-and-project setting and also show that the max-distance rule leads to a convergence guarantee that is *strictly* faster than that of any fixed probability distribution.

2.2.3 The capped adaptive rule

A new family of adaptive sampling methods were recently proposed for the Kaczmarz and coordinate descent type methods [BW18a, BW18b, BW19a]. We extend these methods to the sketch-and-project setting, which allows their application in other settings such as for coordinate descent. While introduced under the names greedy randomized Kaczmarz [BW18a] and relaxed greedy randomized Kaczmarz [BW18b], we rename them here to prevent confusion with the max-distance rule. We instead refer to such methods as *capped adaptive* methods because they select indices i whose corresponding sketched losses $f_i(x^k)$ are larger than a capped threshold given by a convex combination of the largest and mean

sketched losses. It was proven in [BW18a] that the convergence guarantee when using the capped adaptive rule is strictly faster than the fixed non-uniform sampling rule given in [SV09] in the randomized Kaczmarz setting. In Subsection 2.7.5, we generalize this capped adaptive sampling to sketch-and-project methods and prove that the resulting convergence guarantee of this adaptive rule is slower than that of the max-distance rule. Furthermore, in Subsection 2.A.3, we show that the max-distance rule requires less computation at each iteration than the capped adaptive rule.

2.2.4 The proportional adaptive rule

We also present a new and much simpler randomized adaptive rule as compared to the capped adaptive rule discussed above, in which indices are sampled with probabilities that are directly *proportional* to their corresponding sketched losses $f_i(x^k)$. We show that this rule gives a resulting convergence that is strictly faster as when sampling the sketches uniformly.

2.2.5 Efficient implementations

Our adaptive methods come with the added cost of computing the sketched loss $f(x^k)$ of Equation (2.8) at each iteration. Fortunately, the sketched loss can be computed efficiently with certain precomputations as discussed in Section 2.8. We show how the sketched losses can be maintained efficiently via an auxiliary update, leading to reasonably efficient implementations of the adaptive sampling rules. We demonstrate improved performance of the adaptive methods over uniform sampling when solving linear systems with both real and synthetic matrices per iteration and in terms of the *floating point operations (flops)* required.

2.2.6 Consequences and future work

Our results on adaptive sampling are expected to hold analogously many other closely related problems. For instance, a sampling strategy similar to our proportional adaptive rule has

been proposed for coordinate descent in the primal-dual setting for optimizing regularized loss functions [PCJ17]. Also a variant of adaptive and greedy coordinate descent has been shown to speed-up the solution of the matrix scaling problem [AG18]. The matrix scaling problem is equivalent to an entropy-regularized version of the optimal transport problem which has numerous applications in machine learning and computer vision [AG18, Cut13]. Thus the adaptive methods proposed here may be extended to these other settings such as adaptive coordinate descent for more general smooth optimization [PCJ17]. The adaptive methods and the analysis proposed in this chapter may also provide insights toward adaptive sampling for other classes of optimization methods such as stochastic gradient, since the randomized Kaczmarz method can be reformulated as stochastic gradient descent applied to the least-squares problem [NSW16].

2.3 Notation

We now introduce notation that will be used throughout. Let Δ_q denote the simplex in $\mathbb{R}^q$

$$\Delta_q \overset{\text{def}}{=} \{p \in \mathbb{R}^q \ : \ \sum_{i=1}^{q} p_i = 1, \ p_i \geq 0, \ \text{for } i = 1, \ldots, q\}.$$

For probabilities $p \in \Delta_q$ and values a_i depending on an index $i = 1, \ldots, q$, we denote $\mathbb{E}_{i \sim p}[a_i] \overset{\text{def}}{=} \sum_{i=1}^{q} p_i a_i$, where $i \sim p$ indicates that i is sampled with probability p_i. At the k^{th} iteration of the sketch-and-project method, a sketching matrix $\mathbf{S}_{i_k}$ is sampled with probability

$$\mathbb{P}(\mathbf{S}_{i_k} = \mathbf{S}_i \mid x^k) = p_i^k, \quad \text{for } i = 1, \ldots, q, \tag{2.9}$$

where $p^k \in \Delta_q$ and we use $p^k \overset{\text{def}}{=} (p_1^k, \ldots, p_q^k)$ to denote the vector containing these probabilities. We drop the superscript k when the probabilities do not depend on the iteration.

For any symmetric positive semi-definite matrix $\mathbf{G}$ we write the semi-norm induced by $\mathbf{G}$ as $\|\cdot\|_{\mathbf{G}}^2 \overset{\text{def}}{=} \langle \cdot, \mathbf{G}\cdot \rangle$, where $\|\cdot\|$ denotes the standard 2-norm ($\|\cdot\|_2$). For any matrix $\mathbf{M}$, $\|\mathbf{M}\|_F \overset{\text{def}}{=} \sqrt{\sum_{i,j} \mathbf{M}_{ij}^2}$. We use

$$\lambda_{\min}^{+}(\mathbf{G}) \overset{\text{def}}{=} \min_{v \in \text{Range}(\mathbf{G}), v \neq 0} \frac{\|v\|_{\mathbf{G}}^2}{\|v\|_2^2}$$

to denote the smallest non-zero eigenvalue of $\mathbf{G}$.

2.3.1 Organization

The remainder of this chapter is organized as follows. Sections 2.4 and 2.5 provide additional background on the sketch-and-project method and motivation for adaptive sampling in this setting. Section 2.4 explains how the sketch-and-project method can be reformulated as stochastic gradient descent. The sampling of the sketches can then be seen as importance sampling in the context of stochastic gradient descent. Section 2.5 provides geometric intuition for the sketch-and-project method and motivates why one would expect adaptive sampling strategies that depend on the sketched losses $f_i(x^k)$ to perform well.

Section 2.6 introduces the various sketch selection strategies considered throughout the chapter, while Section 2.7 provides convergence guarantees for each of the resulting methods. In Section 2.8, we discuss the computational costs of adaptive sketch-and-project for the sketch selection strategies of Section 2.6 and suggest efficient implementations of the methods. Section 2.9 discusses convergence and computational cost for the special subcases of randomized Kaczmarz and coordinate descent. Performance of adaptive sketch-and-project methods are demonstrated in Section 2.10 for both synthetic and real-world matrices.

2.4 Reformulation as importance sampling for SGD

The sketch-and-project method can be reformulated as a stochastic gradient method, as shown in [RT20]. We use this reformulation to motivate our adaptive sampling as a variant of importance sampling.

Let $p \in \Delta_q$. Consider the stochastic program

$$\min_{x \in \mathbb{R}^d} F(x) \overset{\text{def}}{=} \mathbb{E}_{i \sim p}\left[f_i(x)\right] = \mathbb{E}_{i \sim p}\left[\|\mathbf{A}x - b\|^2_{\mathbf{H}_i}\right]. \tag{2.10}$$

Objective functions $F(x)$ such as the one in Equation (2.10) are common in machine learning, where $f_i(x)$ often represents error of a model with parameters x with respect to datapoint i.

When $\mathbb{E}_{i\sim p}[\mathbf{H}_i]$ is invertible, solving Equation (2.10) is equivalent to solving the linear system Equation (2.1). This invertibility condition on $\mathbb{E}_{i\sim p}[\mathbf{H}_i]$ can be significantly relaxed by using the following technical exactness assumption on the probability p and the set of sketches introduced in [RT20].

Assumption 1. *Let $p \in \Delta_q$, $\Sigma \overset{def}{=} \{S_1, \ldots, S_q\}$ be a set of sketching matrices and $\mathbf{H}_i$ as defined in Equation* (2.7)*. We say that the* exactness assumption *holds for (p, Σ) if*

$$\mathrm{Null}\left(\mathbb{E}_{i\sim p}[\mathbf{H}_i]\right) \subset \mathrm{Null}\left(\mathbf{A}^\top\right).$$

This exactness assumption guarantees[†] that

$$\mathrm{Null}\left(\mathbf{A}\right) = \mathrm{Null}\left(\mathbf{A}^\top \mathbb{E}_{i\sim p}[\mathbf{H}_i]\,\mathbf{A}\right). \tag{2.11}$$

This in turn guarantees that the expected sketched loss of the point x is zero if and only if $\mathbf{A}x = b$. Indeed, by taking the gradient of (2.10) and setting it to zero we have that

$$\nabla F(x) \;=\; \mathbf{A}^\top \mathbb{E}_{i\sim p}[\mathbf{H}_i]\,(\mathbf{A}x - b) \;=\; \mathbf{A}^\top \mathbb{E}_{i\sim p}[\mathbf{H}_i]\,\mathbf{A}(x - x^\star) \;=\; 0.$$

By the convexity of F, we know that any extrema x must be a minimizer. So we see that any minimizer x of Equation (2.10) satisfies

$$x - x^\star \;\in\; \mathrm{Null}\left(\mathbf{A}^\top \mathbb{E}_{i\sim p}[\mathbf{H}_i]\,\mathbf{A}\right) \overset{(2.11)}{=} \mathrm{Null}\left(\mathbf{A}\right), \tag{2.12}$$

and thus $\mathbf{A}(x - x^\star) = \mathbf{A}x - b = 0$. As shown in [GR15a] and [RT20] this exactness assumption holds trivially under modest assumptions.

When the number of f_i functions is too large to afford computing $\nabla F(x)$, the SGD (stochastic gradient descent) method is typically the method of choice for solving Equation (2.10) [BCN18]. To view the sketch-and-project update in Equation (2.6) as a SGD method, we sample an index $i_k \sim p$ at each iteration and take a step

$$x^{k+1} = x^k - \nabla^{\mathbf{B}} f_{i_k}(x^k), \tag{2.13}$$

[†]This can be shown by applying Lemma 2.B.1 in Section 2.B with with $\mathbf{G} = \mathbb{E}_{i\sim p}[\mathbf{H}_i]$ and $\mathbf{W} = \mathbf{A}$.

where $\nabla^{\mathbf{B}} f_{i_k}(x^k)$ is the gradient taken with respect to the $\mathbf{B}$–norm. For $f_i(x^k)$ of Equation (2.8), the exact expression of this stochastic gradient is given by

$$\nabla^{\mathbf{B}} f_{i_k}(x^k) = \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b). \tag{2.14}$$

By plugging Equation (2.14) into Equation (2.13) we can see that the resulting update is equivalent to a the sketch-and-project update in Equation (2.6).

Though the indices $i \in \{1, \ldots, q\}$ are often sampled uniformly at random for SGD, many alternative sampling distributions have been proposed in order to accelerate convergence, including adaptive sampling strategies [CR18, JZ13, NSW16, ZZ15, KF18, LH15, ALS15]. Such sampling strategies give more weight to sampling indices corresponding to a larger loss $f_i(x)$ or a larger gradient norm $\|\nabla^{\mathbf{B}} f_i(x)\|_{\mathbf{B}}^2$. In the sketch-and-project setting, these two sampling strategies result in similar methods since

$$f_i(x) = \|\mathbf{A}x - b\|_{\mathbf{H}_i}^2 = \|\nabla^{\mathbf{B}} f_i(x)\|_{\mathbf{B}}^2$$

by Lemma 3.1 of [RT20].

In general, updating the loss and gradient of every $f_i(x)$ at each iteration can be too expensive. Thus many methods resort to using global approximations of these values such as the Lipschitz constant of the gradient [NSW16] that lead to fixed data-dependent sample distributions. For the sketch-and-project setting, we demonstrate in Section 2.8 that the adaptive sample distributions can be calculated efficiently, with a per-iterate cost on the same order as is required for the sketch-and-project update.

2.5 Geometric viewpoint and motivational analysis

The sketch-and-project method given in Equation (2.5) can be seen as a method that calculates the next iterate x^{k+1} by projecting the previous iterate x^k onto an affine space. Indeed, the constraint in Equation (2.5) can be re-written as

$$\{x \ : \ \mathbf{S}_i^\top \mathbf{A}x = \mathbf{S}_i^\top b\} \quad = \quad x^\star + \text{Null}\left(\mathbf{S}_i^\top \mathbf{A}\right). \tag{2.15}$$

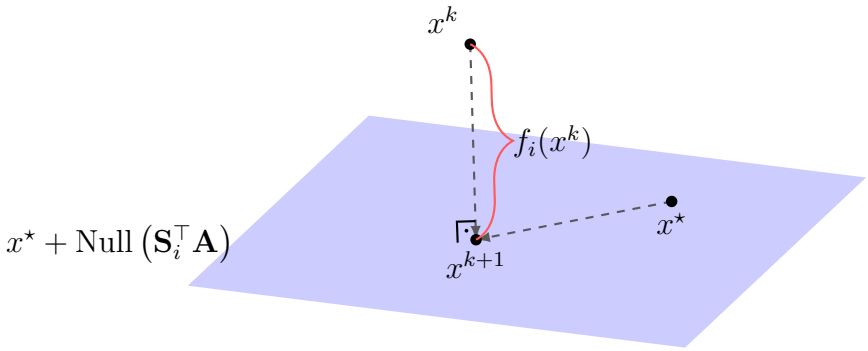


Figure 2.1: The geometric interpretation of Equation (2.5), as the projection of x^k onto an affine space that contains $x^\star$. The distance traveled is given by $f_i(x^k) = \|x^{k+1} - x^k\|^2_{\mathbf{B}}$.

In particular, Equation (2.5) is an orthogonal projection of the point x^k onto an affine space that contains $x^\star$ with respect to the $\mathbf{B}$–norm. See Figure 2.1 for an illustration. This projection is determined by the following projection operator.

Lemma 2.5.1. *Let*

$$\mathbf{Z}_i \overset{def}{=} \mathbf{B}^{-1/2}\mathbf{A}^\top \mathbf{S}_i (\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1/2} = \mathbf{B}^{-1/2}\mathbf{A}^\top \mathbf{H}_i \mathbf{A}\mathbf{B}^{-1/2}, \tag{2.16}$$

for $i = 1, \ldots, q$, which is the orthogonal projection matrix onto $\text{Range}\left(\mathbf{B}^{-1/2}\mathbf{A}^\top \mathbf{S}_i\right)$. *Consequently*

$$\mathbf{Z}_i\mathbf{Z}_i = \mathbf{Z}_i, \quad \textit{and} \quad (\mathbf{I} - \mathbf{Z}_i)\mathbf{Z}_i = 0. \tag{2.17}$$

Furthermore we have that $(\mathbf{I} - \mathbf{Z}_i)$ gives the projection depicted in Figure 2.1 since

$$\mathbf{B}^{1/2}(x^{k+1} - x^\star) \;=\; (\mathbf{I} - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^\star). \tag{2.18}$$

Finally we can re-write the sketched loss as

$$f_i(x) \;=\; \|\mathbf{B}^{1/2}(x - x^\star)\|^2_{\mathbf{Z}_i}, \quad \textit{for } i = 1, \ldots, q. \tag{2.19}$$

Proof. The proof of Equation (2.17) relies on standard properties of the pseudoinverse and is given in Lemma 2.2 in [GR15b].

As for the proof of Equation (2.18), subtracting $x^\star$ from both sides of Equation (2.6) we have that

$$\begin{aligned} x^{k+1} - x^\star &= x^k - x^\star - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b) \\ &\overset{\mathbf{A}x^\star = b}{=} x^k - x^\star - \mathbf{B}^{-1/2}\mathbf{B}^{-1/2}\mathbf{A}^\top \mathbf{H}_{i_k}\mathbf{A}\mathbf{B}^{-1/2}\mathbf{B}^{1/2}(x^k - x^\star) \\ &\overset{(2.16)}{=} x^k - x^\star - \mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^\star). \end{aligned} \tag{2.20}$$

It now only remains to multiply both sides by $\mathbf{B}^{1/2}$.

Finally the proof of Equation (2.19) follows by using $\mathbf{A}x^\star = b$ together with the definitions of $\mathbf{H}_i$ and $\mathbf{Z}_i$ given in Equation (2.7) and Equation (2.16) so that

$$f_i(x) = \|\mathbf{A}(x - x^\star)\|^2_{\mathbf{H}_i} = \|x - x^\star\|^2_{\mathbf{A}^\top \mathbf{H}_i \mathbf{A}} \overset{(2.16)}{=} \|\mathbf{B}^{1/2}(x - x^\star)\|^2_{\mathbf{Z}_i}. \tag{2.21}$$

□

With the explicit expression for the projection operator we can calculate the progress made by a single iteration of the sketch-and-progress method. The convergence proofs in Section 2.7 will rely heavily on Lemmas 2.5.2 and 2.5.3.

Lemma 2.5.2. *Let $x^k \in \mathbb{R}^d$ and let x^{k+1} be given by Equation (2.5). Then the squared magnitude of the update is*

$$\|x^{k+1} - x^k\|^2_{\mathbf{B}} = f_{i_k}(x^k), \tag{2.22}$$

and the error from one iteration to the next decreases according to

$$\|x^{k+1} - x^\star\|^2_{\mathbf{B}} = \|x^k - x^\star\|^2_{\mathbf{B}} - f_{i_k}(x^k). \tag{2.23}$$

Proof. We begin by deriving Equation (2.23). Taking the squared norm in Equation (2.18)

we have

$$\begin{aligned}
\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 &= \|(\mathbf{I} - \mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2})(x^k - x^\star)\|_{\mathbf{B}}^2 \\
&= \|(\mathbf{I} - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^\star)\|_2^2 \\
&= \left\langle \mathbf{B}^{1/2}(x^k - x^\star), (I - \mathbf{Z}_{i_k})(I - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^\star)\right\rangle \\
&\overset{(2.17)}{=} \left\langle \mathbf{B}^{1/2}(x^k - x^\star), (I - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^\star)\right\rangle \\
&= \|x^k - x^\star\|_{\mathbf{B}}^2 - \left\langle \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^\star), \mathbf{B}^{1/2}(x^k - x^\star)\right\rangle \\
&\overset{(2.19)}{=} \|x^k - x^\star\|_{\mathbf{B}}^2 - f_i(x^k). \qquad (2.24)
\end{aligned}$$

Finally we establish Equation (2.22) by subtracting x^k from both sides of Equation (2.6) so that

$$x^{k+1} - x^k = -\mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^\star).$$

It now remains to take the squared $\mathbf{B}$–norm and use Equation (2.19). □

Equation (2.22) shows that the distance traveled from x^k to x^{k+1} is given by the sketch residual $f_{i_k}(x^k)$, as we have depicted in Figure 2.1. Furthermore, Equation (2.23) shows that the contraction of the error $x^{k+1} - x^\star$ is given by $-f_{i_k}(x^k)$. Consequently Lemma 2.5.2 indicates that in order to make the most progress in one step, or maximize the distance traveled, we should choose i_k corresponding to the largest sketched loss $f_{i_k}(x^k)$. We refer to this greedy sketch selection as the *max-distance rule*, which we explore in detail in Subsection 2.6.3.

Next we give the expected decrease in the error.

Lemma 2.5.3. *Let $p^k \in \Delta_q$. Consider the iterates of the sketch-and-project method given in Equation* (2.6) *where $i_k \sim p_i^k$ as is done in Algorithm 2. It follows that*

$$\mathbb{E}_{i\sim p^k}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 \mid x^k\right] = \|x^k - x^\star\|_{\mathbf{B}}^2 - \mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right].$$

Proof. The result follows by taking the expectation over Equation (2.23) conditioned on x^k. □

Lemma 2.5.3 suggests choosing adaptive probabilities so that $\mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right]$ is large. This analysis motivates the adaptive methods described in Subsection 2.6.2.

2.6 Selection rules

Motivated by Lemmas 2.5.2 and 2.5.3, we might think that sampling rules that prioritize larger entries of the sketched loss should converge faster. From this point we take two alternatives, (1) choose the i_k that maximizes the decrease (Subsection 2.6.3) or (2) choose a probability distribution that prioritizes the biggest decrease (Subsection 2.6.2). Below, we describe several sketch-and-project sampling strategies (fixed, adaptive, and greedy) and analyze their convergence in Section 2.7. The adaptive and greedy sampling strategies require knowledge of the current sketched loss vector at each iteration. Calculating the sketched loss from scratch is expensive, thus in Section 2.8 we will show how to efficiently calculate the new sketched loss $f(x^{k+1})$ using the previous sketched loss $f(x^k)$.

2.6.1 Fixed sampling

We first recall the standard non-adaptive sketch-and-project method that will be used as a comparison for the greedy and adaptive versions [GR15b]. In the non-adaptive setting the sketching matrices are sampled from a fixed distribution that is independent of the current iterate x^k. For reference, the details of the non-adaptive sketch-and-project method are provided in Algorithm 1.

2.6.2 Adaptive probabilities

Equation (2.23) motivates selecting indices that correspond to larger sketched losses with higher probability. We refer to such sampling strategies as *adaptive sampling strategies*, as they depend on the current iterate and its corresponding sketched loss values. In the adaptive setting, we sample indices at the k^{th} iteration with probabilities given by $p^k \in \Delta_q$. Adaptive

Algorithm 1 Non-Adaptive Sketch-and-Project

input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $p \in \Delta_q$, and a set of sketching matrices $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_q]$

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

  $i_k \sim p_i$

  $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$

output: last iterate x^{k+1}

sketch-and-project is detailed in Algorithm 2.

Algorithm 2 Adaptive Sketch-and-Project

input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and a set of sketching matrices $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_q]$

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

  $f_i(x^k) = \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}$ for $i = 1, \ldots, q$

  Calculate $p^k \in \Delta_q$ ▷ Typically based on $f(x^k)$.

  $i_k \sim p^k_i$

  $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$

output: last iterate x^{k+1}

2.6.3 Max-distance rule

We refer to the greedy sketch selection rule given by

$$i_k \in \arg\max_{i=1,\ldots,q} f_i(x^k) = \arg\max_{i=1,\ldots,q} \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}, \tag{2.25}$$

as the max-distance selection rule. If multiple indices lead to the maximal sketched loss, any of these indices can be chosen. The max-distance rule leads to the best expected decrease in mean squared error per iteration. The max-distance sketch-and-project method is described in Algorithm 3. This greedy selection strategy has been studied for several specific choices of $\mathbf{B}$ and sketching methods. For example, in the Kaczmarz setting, this strategy is typically referred to as max-distance Kaczmarz or Motzkin's method [GO12, NSL16, MS54]. For

coordinate descent, this selection strategy is the Gauss–Southwell rule [Nes12, NSL15]. We provide a convergence analysis for the general sketch-and-project max-distance selection rule in Theorem 2.7.5. We also show that max-distance selection leads to a convergence rate that is strictly larger than the resulting convergence rate when sampling from any fixed distribution in Theorem 2.7.7.

Algorithm 3 Max-Distance Sketch-and-Project

input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and a set of sketching matrices $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_q]$

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

  $f_i(x^k) = \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}$ for $i = 1, \ldots, q$

  $i_k = \arg\max_{i=1,\ldots,q} f_i(x^k)$

  $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$

output: last iterate x^{k+1}

2.7 Convergence

We now present convergence results for the max-distance selection rule, uniform sampling, and adaptive sampling with probabilities proportional to the sketched loss. We summarize the convergence guarantees discussed throughout Section 2.7 in Table 2.1. Note that these are worst case convergence guarantees and thus are not guaranteed to reflect the expected performance of each selection rule. Our first step in the analysis is to establish an invariance property of the iterates in the following lemma.[‡] In particular, Lemma 2.7.1 guarantees the error vectors $x^k - x^\star$ remain in the subspace Range $\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$ for all iterations if $x^0 \in$ Range $\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$, which allows a tighter convergence analysis.

Lemma 2.7.1. *If* $x^0 \in$ Range $\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$ *then* $x^k - x^\star \in$ Range $\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$.

Proof. First note that $x^\star \in$ Range $\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$. This follows by taking the Lagrangian of

[‡]This lemma was first presented in [GR15a]. We present and prove it here for completeness.

Equation (2.2) given by

$$L(x,\lambda) = \tfrac{1}{2}\|x\|_{\mathbf{B}}^2 + \langle \lambda, \mathbf{A}x - b\rangle\,.$$

Taking the derivative with respect to x, setting to zero and isolating x gives

$$x^\star = -\mathbf{B}^{-1}\mathbf{A}^\top\lambda \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right). \tag{2.26}$$

Consequently $x^\star - x^0 \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$. Assuming that $x^k - x^\star \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$ holds, by induction we have that

$$x^{k+1} - x^\star \overset{(2.6)}{=} x^k - x^\star - \underbrace{\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}(\mathbf{S}_{i_k}^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k})^\dagger\mathbf{S}_{i_k}^\top(\mathbf{A}x^k - b)}_{\in\text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)}. \tag{2.27}$$

Thus $x^{k+1} - x^\star$ is the difference of two elements in the subspace Range $\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$ and thus $x^{k+1} - x^\star \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$. □

We also make use of the following fact. For a symmetric positive semi-definite random matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ drawn from some probability distribution $\mathcal{D}$ and for any vector $v \in \mathbb{R}^n$

$$\mathbb{E}_{\mathcal{D}}\left[\|v\|_{\mathbf{M}}^2\right] = \mathbb{E}_{\mathcal{D}}\left[\langle v, \mathbf{M}v\rangle\right] = \langle v, \mathbb{E}_{\mathcal{D}}\left[\mathbf{M}v\right]\rangle = \|v\|_{\mathbb{E}_{\mathcal{D}}[\mathbf{M}]}^2. \tag{2.28}$$

2.7.1 Important spectral constants

We define two key spectral constants in the following definition that will be used to express our forthcoming rates of convergence.

Definition 2.7.1 (Spectral Constants)**.** *The spectral constant for max-distance methods is*

$$\sigma_\infty^2(\mathbf{B},\mathbf{S}) \overset{def}{=} \min_{v\in\text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|_{\mathbf{Z}_i}^2}{\|v\|_{\mathbf{B}}^2}, \tag{2.29}$$

while the spectral constant for a fixed distribution $p \in \Delta_q$ is

$$\sigma_p^2(\mathbf{B},\mathbf{S}) \overset{def}{=} \min_{v\in\text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \frac{\|\mathbf{B}^{1/2}v\|_{\mathbb{E}_{i\sim p}[\mathbf{Z}_i]}^2}{\|v\|_{\mathbf{B}}^2}. \tag{2.30}$$

Next we show that $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ and $\sigma_p^2(\mathbf{B},\mathbf{S})$ can be used to lower bound $\max_i f_i(x)$ and $\mathbb{E}_{i\sim p}\left[f_i(x)\right]$, respectively. This result will allow us to develop Equation (2.23) and Lemma 2.5.3 into recurrences used to prove the Theorems in the remainder of this section.

Lemma 2.7.2. *Let $p \in \Delta_q$ and consider the iterates x^k given by Algorithm 2. The spectral constants Equation* (2.29) *and Equation* (2.30) *are such that*

$$\max_{i=1,\ldots,q} f_i(x^k) \quad \geq \quad \sigma_\infty^2(\mathbf{B},\mathbf{S})\|x^k - x^\star\|_{\mathbf{B}}^2, \tag{2.31}$$

$$\mathbb{E}_{i\sim p}\left[f_i(x^k)\right] \quad \geq \quad \sigma_p^2(\mathbf{B},\mathbf{S})\|x^k - x^\star\|_{\mathbf{B}}^2. \tag{2.32}$$

Proof. From the invariance provided by Lemma 2.7.1 we have that $x^k - x^\star \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$ and consequently

$$\begin{aligned}\frac{\max_{i=1,\ldots,q} f_i(x^k)}{\|x^k - x^\star\|_{\mathbf{B}}^2} &\overset{(2.19)}{=} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}(x^k - x^\star)\|_{\mathbf{Z}_i}^2}{\|x^k - x^\star\|_{\mathbf{B}}^2} \\ &\geq \min_{v\in\text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|_{\mathbf{Z}_i}}{\|v\|_{\mathbf{B}}^2} \overset{(2.29)}{=} \sigma_\infty^2(\mathbf{B},\mathbf{S}), \quad \forall k. \end{aligned} \tag{2.33}$$

Analogously we have that

$$\begin{aligned}\frac{\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]}{\|x^k - x^\star\|_{\mathbf{B}}^2} &\overset{(2.19)}{=} \frac{\mathbb{E}_{i\sim p}\left[\|\mathbf{B}^{1/2}(x^k - x^\star)\|_{\mathbf{Z}_i}^2\right]}{\|x^k - x^\star\|_{\mathbf{B}}^2} \\ &\geq \min_{v\in\text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \frac{\mathbb{E}_{i\sim p}\left[\|\mathbf{B}^{1/2}v\|_{\mathbf{Z}_i}^2\right]}{\|v\|_{\mathbf{B}}^2} \overset{(2.30)+(2.28)}{=} \sigma_p^2(\mathbf{B},\mathbf{S}), \quad \forall k. \end{aligned} \tag{2.34}$$

Thus Equation (2.31) and Equation (2.32) follow by re-arranging Equation (2.33) and Equation (2.34) respectively. □

Finally, we show that $\sigma_p^2(\mathbf{B},\mathbf{S}), \sigma_\infty^2(\mathbf{B},\mathbf{S}) \leq 1$, and if the exactness Assumption 1 holds then additionally $\sigma_p^2(\mathbf{B},\mathbf{S}), \sigma_\infty^2(\mathbf{B},\mathbf{S}) > 0$.

Lemma 2.7.3. *Let $p \in \Delta_q$ and the set of sketching matrices $\{\mathbf{S}_1,\ldots,\mathbf{S}_q\}$ be such that that exactness Assumption 1 holds. We then have the following relations:*

$$0 < \sigma_p^2(\mathbf{B},\mathbf{S}) \quad = \quad \lambda_{\min}^+\left(\mathbf{E}_{i\sim p}\left[\mathbf{Z}_i\right]\right) \quad \leq \quad \sigma_\infty^2(\mathbf{B},\mathbf{S}) \leq 1.$$

Proof. Using the definition of $\mathbf{Z}_i$ given in Equation (2.16) and the fact that $\mathbf{B}$ is symmetric positive definite, we have

$$\begin{aligned}\operatorname{Null}\left(\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]\right) &\overset{(2.16)}{=} \operatorname{Null}\left(\mathbf{B}^{-1/2}\mathbf{A}^\top\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}\mathbf{B}^{-1/2}\right)\\ &= \operatorname{Null}\left(\mathbf{A}^\top\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}\mathbf{B}^{-1/2}\right) \overset{Lemma\ 2.B.1}{=} \operatorname{Null}\left(\mathbf{A}\mathbf{B}^{-1/2}\right),\end{aligned}$$

where we applied Lemma 2.B.1 with $\mathbf{G} = \mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]$ and $\mathbf{W} = \mathbf{A}$. Taking the orthogonal complement of the above we have that

$$\operatorname{Range}\left(\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]\right) = \operatorname{Range}\left(\mathbf{B}^{-1/2}\mathbf{A}^\top\right). \tag{2.35}$$

Using Equations (2.30) and (2.35) we then have

$$\begin{aligned}\sigma_p^2(\mathbf{B},\mathbf{S}) &\overset{(2.30)}{=} \min_{v\in\operatorname{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{E}_{i\sim p}[\mathbf{Z}_i]}}{\|v\|^2_{\mathbf{B}}}\\ &\overset{(2.35)}{=} \min_{\mathbf{B}^{1/2}v\in\operatorname{Range}(\mathbb{E}_{i\sim p}[\mathbf{Z}_i])} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{E}_{i\sim p}[\mathbf{Z}_i]}}{\|v\|^2_{\mathbf{B}}} \;=\; \lambda^+_{\min}\left(\mathbf{E}_{i\sim p}\left[\mathbf{Z}_i\right]\right) > 0.\end{aligned}$$

Furthermore,

$$\begin{aligned}\sigma_p^2(\mathbf{B},\mathbf{S}) &\overset{(2.30)}{=} \min_{v\in\operatorname{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{E}_{i\sim p}[\mathbf{Z}_i]}}{\|v\|^2_{\mathbf{B}}}\\ &\overset{(2.28)}{=} \min_{v\in\operatorname{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \frac{\mathbf{E}_{i\sim p}\left[\|\mathbf{B}^{1/2}v\|^2_{\mathbf{Z}_i}\right]}{\|v\|^2_{\mathbf{B}}}\\ &\leq \min_{v\in\operatorname{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{Z}_i}}{\|v\|^2_{\mathbf{B}}} \;=\; \sigma_\infty^2(\mathbf{B},\mathbf{S}).\end{aligned}$$

Finally, using the fact that the matrix $\mathbf{Z}_i$ is an orthogonal projection (Lemma 2.5.1), we have that

$$\sigma_\infty^2(\mathbf{B},\mathbf{S}) \;=\; \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{Z}_i}}{\|v\|^2_{\mathbf{B}}} \overset{(2.17)}{=} \max_{i=1,\ldots,q} \frac{\|\mathbf{Z}_i\mathbf{B}^{1/2}v\|^2}{\|\mathbf{B}^{1/2}v\|^2} \;\leq\; \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|^2}{\|\mathbf{B}^{1/2}v\|^2} = 1.$$

□

2.7.2 Sampling from a fixed distribution

We first present a convergence result for the sketch-and-project method when the sketches are drawn from a fixed sampling distribution. This result will be used as a baseline for comparison against the adaptive sampling strategies in Subsections 2.7.3 to 2.7.5.

Theorem 2.7.4. *Consider Algorithm 1 for some set of probabilities $p \in \Delta_q$. It follows that*

$$\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right] \leq \left(1 - \sigma_p^2(\mathbf{B}, \mathbf{S})\right)^k \|x^0 - x^\star\|_{\mathbf{B}}^2.$$

Proof. Combining Lemma 2.5.3 and Equation (2.32) of Lemma 2.7.2 we have that

$$\begin{aligned} \mathbb{E}_{i_k \sim p}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 \mid x^k\right] &\overset{Lemma\ 2.5.3}{=} \|x^k - x^\star\|_{\mathbf{B}}^2 - \mathbb{E}_{i_k \sim p}\left[f_i(x^k)\right] \\ &\overset{(2.32)}{\leq} \left(1 - \sigma_p^2(\mathbf{B}, \mathbf{S})\right) \|x^k - x^\star\|_{\mathbf{B}}^2. \end{aligned}$$

Taking the unconditional expectation and unrolling the recurrence, we arrive at the desired result

$$\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right] \leq \left(1 - \sigma_p^2(\mathbf{B}, \mathbf{S})\right) \mathbb{E}\left[\|x^{k-1} - x^\star\|_{\mathbf{B}}^2\right] \leq \ldots \leq \left(1 - \sigma_p^2(\mathbf{B}, \mathbf{S})\right)^k \|x^0 - x^\star\|_{\mathbf{B}}^2.$$

□

There are several natural and previously studied choices for fixed sampling distributions, for example, sampling the indices uniformly at random. Another choice is to pick $p \in \Delta_q$ in order to maximize $\sigma_p^2(\mathbf{B}, \mathbf{S})$, but this results in a convex semi-definite program (see Section 5.1 in [GR15b]). The authors of [GR15b] suggest convenient probabilities such that $p_i \propto \|\mathbf{A}^\top \mathbf{S}_i\|_{\mathbf{B}^{-1}}^2$ for which $\sigma_p^2(\mathbf{B}, \mathbf{S})$ reduces to the scaled condition number.

2.7.3 Max-distance selection

The following theorem provides a convergence guarantee for the max-distance selection rule of Subsection 2.6.3. To our knowledge, this is the first analysis of the max-distance rule for general sketch-and-project methods.

Theorem 2.7.5. *The iterates of max-distance sketch-and-project method in Algorithm 3 satisfy*

$$\|x^k - x^\star\|_{\mathbf{B}}^2 \leq (1 - \sigma_\infty^2(\mathbf{B}, \mathbf{S}))^k \|x^0 - x^\star\|_{\mathbf{B}}^2,$$

where $\sigma_\infty(\mathbf{B}, \mathbf{S})$ *is defined as in Equation* (2.29) *of Definition 2.7.1.*

Proof. Combining Equation (2.23) and Equation (2.31) we have that

$$\begin{aligned} \|x^{k+1} - x^\star\|_{\mathbf{B}}^2 &\overset{(2.23)}{=} \|x^k - x^\star\|_{\mathbf{B}}^2 - \max_{i=1,\ldots,q} f_i(x^k) \\ &\overset{(2.31)}{\leq} \left(1 - \sigma_\infty^2(\mathbf{B}, \mathbf{S})\right) \|x^k - x^\star\|_{\mathbf{B}}^2. \end{aligned}$$

Unrolling the recurrence gives the desired result

$$\|x^k - x^\star\|_{\mathbf{B}}^2 \leq \left(1 - \sigma_\infty^2(\mathbf{B}, \mathbf{S})\right) \|x^{k-1} - x^\star\|_{\mathbf{B}}^2 \leq \ldots \leq \left(1 - \sigma_\infty^2(\mathbf{B}, \mathbf{S})\right)^k \|x^0 - x^\star\|_{\mathbf{B}}^2.$$

□

One obvious disadvantage of sampling from a fixed distribution is that it is possible to sample the same index on consecutive iterations. Since the current iterate already lies in the solution space with respect to the previous sketch, no progress is made in such an update. For adaptive distributions that only assign non-zero probabilities to non-zero sketched loss values, the same index will never be chosen on consecutive iterations since the sketched loss corresponding to the previous iterate will always be zero (Lemma 2.7.6). This fact allows us to derive convergence rates for adaptive sampling strategies that are strictly better than those for fixed sampling strategies.

Lemma 2.7.6. *Consider the sketched losses* $f(x^k)$ *generated by iterating the sketch-and-project update given in Equation* (2.6)*. We have that*

$$f_{i_k}(x^{k+1}) = 0, \quad \forall\, k \geq 0.$$

Proof. Recall from Equation (2.19), we can write

$$f_{i_k}(x^{k+1}) = \|\mathbf{B}^{1/2}(x^{k+1} - x^\star)\|_{\mathbf{Z}_{i_k}}^2 = \left\langle \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^{k+1} - x^\star), \mathbf{B}^{1/2}(x^{k+1} - x^\star) \right\rangle. \tag{2.36}$$

Using Equation (2.18) and Lemma 2.5.1, we can show that the above is equal to zero

$$\begin{aligned}
\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^{k+1}-x^\star) &\overset{(2.18)}{=} \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k-\mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k-x^\star)-x^\star) \\
&= \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k-x^\star)-\mathbf{Z}_{i_k}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k-x^\star)) \\
&\overset{(2.17)}{=} \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k-x^\star)-\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k-x^\star)) \\
&= 0.
\end{aligned}$$

□

We now use Lemma 2.7.6 to additionally show that the convergence guarantee for the greedy method is strictly faster than for sampling with respect to any set of fixed probabilities.

Theorem 2.7.7. *Let $p \in \Delta_q$ where $p_i > 0$ for all $i = 1, \ldots, q$. Let $\sigma_p^2(\mathbf{B}, \mathbf{S})$ be defined as in Equation* (2.30) *of Definition 2.7.1 and define*

$$\gamma \overset{def}{=} \frac{1}{\max_{i=1,\ldots,q} \sum_{j=1,\, j\neq i}^{q} p_j} > 1. \tag{2.37}$$

We then have that the max-distance sketch-and-project method of Algorithm 3 satisfies the following convergence guarantee

$$\|x^{k+1}-x^\star\|_{\mathbf{B}}^2 \leq (1-\gamma\sigma_p^2(\mathbf{B},\mathbf{S}))\|x^k-x^\star\|_{\mathbf{B}}^2. \tag{2.38}$$

Proof. Recall that $f_{i_k}(x^{k+1}) = 0$ by Lemma 2.7.6. Thus,

$$\begin{aligned}
\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right] &= \sum_{j=1,\, j\neq i_k}^{q} p_j f_j(x^{k+1}) \\
&\leq \left(\max_{j=1,\ldots,q} f_j(x^{k+1})\right)\left(\sum_{j=1,\, j\neq i_k}^{q} p_j\right) \\
&\leq \left(\max_{j=1,\ldots,q} f_j(x^{k+1})\right)\left(\max_{j=1,\ldots,q}\sum_{j=1,\, j\neq i}^{q} p_j\right) \\
&\overset{(2.37)}{=} \frac{\max_{j=1,\ldots,q} f_j(x^{k+1})}{\gamma}.
\end{aligned} \tag{2.39}$$

From Equation (2.23) we have that

$$\begin{aligned}\|x^{k+1}-x^\star\|_{\mathbf{B}}^2 &\overset{(2.23)}{=} \|x^k-x^\star\|_{\mathbf{B}}^2 - \max_{i=1,\dots,q} f_i(x^k)\\ &\overset{(2.39)}{\leq} \|x^k-x^\star\|_{\mathbf{B}}^2 - \gamma\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]\\ &\overset{(2.32)}{\leq} \left(1-\gamma\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^\star\|_{\mathbf{B}}^2.\end{aligned}$$

□

2.7.4 The proportional adaptive rule

We now consider the adaptive sampling strategy in which indices are sampled with probabilities proportional to the sketched loss values. For this sampling strategy, we derive a convergence rate that is strictly faster than that of Theorem 2.7.4 for uniform sampling.

Theorem 2.7.8. *Consider Algorithm 2 with* $p^k = \frac{f(x^k)}{\|f(x^k)\|_1}$. *Let* $u = \left(\frac{1}{q},\dots,\frac{1}{q}\right) \in \Delta_q$ *and* $\sigma_u^2(\mathbf{B},\mathbf{S})$ *be as defined in Equation* (2.30). *Let* $\mathbb{VAR}_u[\cdot]$ *denote the variance taken with respect to the uniform distribution over indices* $i \in \{1,\dots,q\}$. *It follows that for* $k \geq 1$,

$$\mathbb{E}\left[\|x^{k+1}-x^\star\|_{\mathbf{B}}^2 \,|\, x^k\right] \leq \left(1-(1+q^2\mathbb{VAR}_u\left[p_i^k\right])\sigma_u^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^\star\|_{\mathbf{B}}^2. \tag{2.40}$$

Furthermore we have that

$$\mathbb{E}\left[\|x^{k+1}-x^\star\|_{\mathbf{B}}^2\right] \leq \left(1-\left(1+\tfrac{1}{q}\right)\sigma_u^2(\mathbf{B},\mathbf{S})\right)^k \mathbb{E}\left[\|x^1-x^\star\|_{\mathbf{B}}^2\right]. \tag{2.41}$$

Proof. Let $\mathbb{E}_u[\cdot]$ denote the expectation taken with respect to the uniform distribution over indices $i \in \{1,\dots,q\}$. First note that

$$\mathbb{VAR}_u\left[f_i(x^k)\right] = \mathbb{E}_u\left[(f_i(x^k))^2\right] - \mathbb{E}_u\left[f_i(x^k)\right]^2 = \frac{1}{q}\sum(f_i(x^k))^2 - \frac{1}{q^2}\left(\sum f_i(x^k)\right)^2. \tag{2.42}$$

Given that $p^k = \frac{f(x^k)}{\|f(x^k)\|_1}$,

$$\begin{aligned}
\mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right] &= \sum_{i=1}^{q} p_i^k f_i(x^k) \\
&= \sum_{i=1}^{q} \frac{(f_i(x^k))^2}{\sum_{i=1}^{q} f_i(x^k)} \\
&\overset{(2.42)}{=} \frac{q\mathbb{VAR}_u\left[f_i(x^k)\right] + \frac{1}{q}\left(\sum f_i(x^k)\right)^2}{\sum_{i=1}^{q} f_i(x^k)} \\
&= \left(q^2\mathbb{VAR}_u\left[\frac{f_i(x^k)}{\sum_{i=1}^{q} f_i(x^k)}\right] + 1\right)\frac{1}{q}\sum_{i=1}^{q} f_i(x^k). \qquad (2.43)
\end{aligned}$$

Recalling that $p_i^k = \frac{f_i(x^k)}{\sum_{i=1}^{q} f_i(x^k)}$ and using Lemma 2.5.3 we have that

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 \,|\, x^k\right] \leq \|x^k - x^\star\|_{\mathbf{B}}^2 - (1 + q^2\mathbb{VAR}_u\left[p_i^k\right])\sigma_u^2(\mathbf{B},\mathbf{S})\|x^k - x^\star\|_{\mathbf{B}}^2.$$

Furthermore, due to Lemma 2.7.1 we have that $p_{i_k}^{k+1} = 0$. Therefore

$$\begin{aligned}
\mathbb{VAR}_u\left[p_i^{k+1}\right] &= \frac{1}{q}\sum_{i=1}^{q}\left(p_i^{k+1} - \frac{1}{q}\sum_{s=1}^{q} p_s^{k+1}\right)^2 \\
&= \frac{1}{q}\sum_{i=1}^{q}\left(p_i^{k+1} - \frac{1}{q}\right)^2 \geq \frac{1}{q}\left(p_{i_k}^{k+1} - \frac{1}{q}\right)^2 = \frac{1}{q^3}.
\end{aligned}$$

This lower bound on the variance gives the following upper bound on Equation (2.40)

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 \,|\, x^k\right] \leq \left(1 - \left(1 + \tfrac{1}{q}\right)\sigma_u^2(\mathbf{B},\mathbf{S})\right)\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right].$$

Taking the expectation and unrolling the recursion gives the desired result

$$\begin{aligned}
\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right] &\leq \left(1 - \left(1 + \tfrac{1}{q}\right)\sigma_u^2(\mathbf{B},\mathbf{S})\right)\|x^{k-1} - x^\star\|_{\mathbf{B}}^2 \\
&\leq \dots \\
&\leq \left(1 - \left(1 + \tfrac{1}{q}\right)\sigma_u^2(\mathbf{B},\mathbf{S})\right)^k \|x^0 - x^\star\|_{\mathbf{B}}^2.
\end{aligned}$$

□

Thus by sampling proportional to the sketched losses the sketch-and-project method enjoys a strictly faster convergence rate as compared to sampling uniformly. How much faster

depends on the variance of the adaptive probabilities through $1 + q^2 \mathbb{VAR}_u \left[p_i^k\right]$ which in turn depends on the variance of the sketched losses.

This same variance term is used in [PCJ17] to analyze the convergence of an adaptive sampling strategy based on the dual residuals for coordinate descent applied to regularized loss functions and in [OAL16] for adaptive sampling in the block-coordinate Frank-Wolfe algorithm for optimizing structured support vector machines.

2.7.5 Capped adaptive sampling

We now extend the capped adaptive sampling method and convergence guarantees of [BW18a, BW18b, BW19a] for the randomized Kaczmarz and coordinate descent settings to the general sketch-and-project setting. The extension is defined in Algorithm 4. At each iteration k an index set $\mathcal{W}_k$ is constructed on Line 4 of Algorithm 4 that contains indices whose sketched losses are sufficiently close to the maximal sketched loss and that are at least as large as $\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]$, where $p \in \Delta_q$ is a fixed reference probability. At each iteration, the adaptive probabilities p_i^k are zero for all indices that are not included in the set $\mathcal{W}_k$. The input parameter $\theta \in [0,\, 1]$ controls how aggressive the sampling method is. In particular, if $\theta = 1$, the method reduces to the max-distance rule. As $\theta \to 0$, the sampling method remains adaptive, as only indices corresponding to sketched losses larger than $\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]$ are sampled with non-zero probability. Bai and Wu originally introduced an adaptive randomized Kaczmarz method with $\theta = 1/2$ [BW18a] and generalized this to allow the more general choice of $\theta \in [0,\, 1]$[BW18b].

Algorithm 4 generalizes and improves upon the methods proposed in [BW18a, BW18b, BW19a] in several ways. We generalize the methods from the randomized Kaczmarz setting to the more general sketch-and-project setting. We additionally allow the use of any fixed reference probability distribution $p \in \Delta_q$, whereas the methods of [BW18a, BW18b, BW19a] use a specific reference probability when identifying the set of indices that will be selected with nonzero probability. Lastly, we allow the use of any adaptive sampling strategy such

that the probabilities p_i^k are zero outside of the set $\mathcal{W}_k$ whereas the methods proposed in [BW18a, BW18b, BW19a] specify that a specific adaptive probability be used. The specific probability is unnecessary in proving the accompanying convergence result Theorem 2.7.10.

Below, we provide two convergence guarantees for Algorithm 4. Theorem 2.7.9 provides a convergence guarantee in terms of the spectral constants $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ and $\sigma_p^2(\mathbf{B},\mathbf{S})$ of Definition 2.7.1 and the parameter θ. Theorem 2.7.10 provides a generalization of the convergence rate derived in [BW18b].

Algorithm 4 Capped Adaptive Sketch-and-Project

input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $p \in \Delta_q$, $\theta \in [0,\,1]$ and a set of sketching matrices $\{\mathbf{S}_1,\ldots,\mathbf{S}_q\}$

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

  $f_i(x^k) = \|\mathbf{A}x^k - b\|_{\mathbf{H}_i}^2$ for $i = 1,\ldots,q$.

  $\mathcal{W}_k = \left\{ i \mid f_i(x^k) \geq \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right] \right\}$

  Choose $p^k \in \Delta_q$ such that $\text{support}(p^k) \subset \mathcal{W}_k$

  $i_k \sim p^k$

  $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$

output: last iterate x^{k+1}

Theorem 2.7.9. *Consider Algorithm 4. Let $p \in \Delta_q$ be a fixed reference probability and $\theta \in [0,1]$. Let*

$$\mathcal{W}_k = \left\{ i \mid f_i(x^k) \geq \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right] \right\}. \tag{2.44}$$

It follows that

$$\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right] \leq \left(1 - \theta\sigma_\infty^2(\mathbf{B},\mathbf{S}) - (1-\theta)\sigma_p^2(\mathbf{B},\mathbf{S})\right)^k \|x^0 - x^\star\|_{\mathbf{B}}^2. \tag{2.45}$$

Proof. First note that $\mathcal{W}_k$ is not empty since

$$\max_{j=1,\ldots,q} f_j(x^k) \geq \mathbb{E}_{j\sim p}\left[f_j(x^k)\right],$$

and thus $\arg\max_{j=1,\ldots,q} f_j(x^k) \in \mathcal{W}_k$. Since $p_i^k = 0$ for all $i \notin \mathcal{W}_k$, Lemma 2.5.3 gives that

$$\mathbb{E}_{i\sim p^k}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 \,|\, x^k\right] = \|x^{k+1} - x^\star\|_{\mathbf{B}}^2 - \sum_{i\in\mathcal{W}_k} p_i^k f_i(x^k). \tag{2.46}$$

We additionally have

$$\begin{aligned}
\sum_{i\in\mathcal{W}_k} f_i(x^k)p_i^k &\overset{(2.44)}{\geq} \sum_{i\in\mathcal{W}_k}\left(\theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]\right)p_i^k \\
&= \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right] \qquad (2.47)\\
&\overset{Lemma\ 2.7.2}{\geq} \left(\theta\sigma_\infty^2(\mathbf{B},\mathbf{S}) + (1-\theta)\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^k - x^\star\|_{\mathbf{B}}^2. \qquad (2.48)
\end{aligned}$$

Using Equation (2.48) to bound Equation (2.46) and taking the expectation gives the result. □

The resulting convergence rate is a convex combination of the spectral constant $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ which corresponds to the max-distance convergence guarantee and $\sigma_p^2(\mathbf{B},\mathbf{S})$ corresponding to the convergence guarantee for the fixed reference probabilities p. This convex combination is in terms of the parameter θ and we can see that as $\theta \to 1$ the method and convergence guarantee approach that of max-distance. When $\theta \to 0$, the convergence guarantee approaches that of a fixed distribution, but still filters out sketches with sketched losses less than $\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]$. This suggests that for $\theta \approx 0$ the convergence guarantee could be improved.

We now explicitly extend the analysis of Bai and Wu [BW18b, BW18b, BW19a] to derive a convergence guarantee for our more general Algorithm 4.

Theorem 2.7.10. *Consider Algorithm 4. Let $p \in \Delta_q$ be a set of fixed reference probabilities and $\theta \in [0,1]$. Let*

$$\gamma \overset{def}{=} \frac{1}{\max_{i=1,\ldots,q}\sum_{j=1,\,j\neq i}^q p_j} > 1. \tag{2.49}$$

It follows for $k \geq 1$

$$\begin{aligned}
&\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right] \qquad (2.50)\\
&\leq \left(1 - (\theta\gamma + (1-\theta))\,\sigma_p^2(\mathbf{B},\mathbf{S})\right)^{k-1}\left(1 - \theta\sigma_\infty^2(\mathbf{B},\mathbf{S}) - (1-\theta)\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^0 - x^\star\|_{\mathbf{B}}^2,
\end{aligned}$$

where the expectation is taken with respect to the probabilities prescribed by Algorithm 4.

Proof. By Lemma 2.7.6, at least one of the sketched losses is guaranteed to be zero for each iterations $k \geq 1$. Making the conservative assumption that this sketched loss corresponds to the smallest probability $\hat{p}_{i_k}^k$, we have, by Equation (2.39), that for an adaptive sampling strategy that assigns $p_i^k = 0$ to sketches $\mathbf{S}_i$ with a sketched loss $f_i(x^k) = 0$ that

$$\frac{\max_{j=1,\dots,q} f_j(x^{k+1})}{\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right]} \geq \gamma. \tag{2.51}$$

Combining this with Equation (2.47), we obtain

$$\begin{aligned}\sum_{i\in\mathcal{W}_k} f_i(x^{k+1})p_i^{k+1} &\geq \left(\theta\frac{\max_{j=1,\dots,q} f_j(x^{k+1})}{\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right]} + (1-\theta)\right)\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right] \\ &\overset{(2.51)}{\geq} \left(\theta\gamma + (1-\theta)\right)\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right] \\ &\overset{(2.30)}{\geq} \left(\theta\gamma + (1-\theta)\right)\sigma_p^2(\mathbf{B},\mathbf{S}). \end{aligned} \tag{2.52}$$

Consequently for $k \geq 1$, by Equation (2.46), we then have

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2 \,|\, x^k\right] \leq \|x^k - x^\star\|_{\mathbf{B}}^2 - \left(\theta\gamma + (1-\theta)\right)\sigma_p^2(\mathbf{B},\mathbf{S})\|x^k - x^\star\|_{\mathbf{B}}^2.$$

Taking the expectation and unrolling the recursion gives

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_{\mathbf{B}}^2\right] \leq \left(1 - \left(\theta\gamma + (1-\theta)\right)\sigma_p^2(\mathbf{B},\mathbf{S})\right)^{k-1}\|x^1 - x^\star\|_{\mathbf{B}}^2.$$

Since, at the very first update, we cannot guarantee that there exists $i \in [1,\dots,q]$ such that $f_i(x^0) = 0$, Equation (2.52) is not guaranteed for $k = 0$. So instead we use Equation (2.45) at the last step in this recurrence to arrive at Equation (2.50). □

The convergence rate for Algorithm 4 of Theorem 2.7.10 is an improvement over the convergence guarantee for a fixed probability distribution since $\gamma > 1$. As was the case for Theorem 2.7.9, the convergence rate is maximized when $\theta = 1$, at which point the resulting method is equivalent to the max-distance rule of Algorithm 3. Further, when $\theta = 1$, Theorem 2.7.10 guarantees

$$\mathbb{E}\left[\|x^k - x^\star\|_{\mathbf{B}}^2\right] \leq \left(1 - \gamma\sigma_p^2(\mathbf{B},\mathbf{S})\right)^{k-1}\left(1 - \sigma_\infty^2(\mathbf{B},\mathbf{S})\right)\|x^0 - x^\star\|_{\mathbf{B}}^2.$$

For $\theta = 0$, Theorem 2.7.10 recovers the same convergence guarantee as for sampling according to the non-adaptive probabilities p.

Sampling strategy	Convergence rate bound	Shown in
Fixed probabilities $p_i^k \equiv p_i$	$1 - \sigma_p^2(\mathbf{B}, \mathbf{S})$	[GR15b], Theorem 2.7.4
Max-distance	$1 - \sigma_\infty^2(\mathbf{B}, \mathbf{S})$	Theorem 2.7.5
$p_i^k \propto f_i(x^k)$	$1 - \left(1 + \frac{1}{q}\right) \sigma_u^2(\mathbf{B}, \mathbf{S})$	Theorem 2.7.8
Capped	$1 - (\theta\gamma + (1-\theta)) \, \sigma_p^2(\mathbf{B}, \mathbf{S})$	Theorem 2.7.10

Table 2.1: Summary of convergence guarantees of Section 2.7.

2.8 Implementation tricks and computational complexity

One can perform adaptive sketching with the same order of cost per iteration as the standard non-adaptive sketch-and-project method when τq, the number of sketches q times the sketch size τ, is not significantly larger than the number of columns n. In particular, adaptive sketching methods can be performed for a per-iteration cost of $O(\tau^2 q + \tau n)$, whereas the standard non-adaptive sketch-and-project method has a per-iteration cost of $O(\tau n)$. Section 2.A discusses the costs of adaptive sketch-and-project methods in more detail. Pseudocode for efficient implementation is provided in Algorithm 5.

The main computational costs of adaptive sketch-and-project (Algorithm 2) at each iteration come from computing the sketched losses $f_i(x^k)$ of Equation (2.8) and updating the iterate from x^k to x^{k+1} via Equation (2.6). The iterate update for x^k and the formula for the sketched loss $f_i(x^k) = \|\mathbf{A}x - b\|_{\mathbf{H}_i}^2$ both require calculating what we call the *sketched residual*,

$$\mathbf{R}_i^k \overset{\text{def}}{=} \mathbf{C}_i^\top \mathbf{S}_i^\top (\mathbf{A}x^k - b), \tag{2.53}$$

where $\mathbf{C}_i$ is any square matrix satisfying $\mathbf{C}_i \mathbf{C}_i^\top = (\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger$. The adaptive methods considered here require the sketched residual $\mathbf{R}_i^k$ for each sketch index $i = 1, 2, \ldots, q$ at each iteration. For such adaptive methods, it is possible to update the iterate x^k and compute the sketched losses $f_i(x^k)$ more efficiently if one maintains the set of sketched residuals $\{\mathbf{R}_i^k : i = 1, 2, \ldots, q\}$ in memory.

Different sampling strategies require different amounts of computation as well. Among the adaptive sampling strategies considered here, max-distance requires the least amount of computation followed by sampling proportional to the sketched losses. Capped adaptive sampling requires the most computation. The costs for each sampling strategy are discussed in detail in Subsection 2.A.3 and are summarized in Tables 2.7 and 2.8.

2.9 Summary of consequences for special cases

We now discuss the consequences of the convergence analyses of Section 2.7 and the computational costs detailed in Section 2.8 for the special sketch-and-project subcases of randomized Kaczmarz and coordinate descent. For $\mathbf{C}_i$ as defined in Equation (2.54), in both the randomized Kaczmarz method and coordinate descent, $\mathbf{C}_i$ is a scalar.

2.9.1 Adaptive Kaczmarz

By choosing the parameter matrix $\mathbf{B} = \mathbf{I}$ and sketching matrices $\mathbf{S}_i = e_i$ for $i = 1, \ldots, m$ where $e_i \in \mathbb{R}^n$ is the i^{th} coordinate vector, we arrive at the Kaczmarz method discussed in Subsection 2.1.1. For randomized Kaczmarz, the sketches $\mathbf{S}_i = e_i$ isolate a single row of the matrix $\mathbf{A}$, as $\mathbf{S}_i^\top \mathbf{A} = \mathbf{A}_{i:}$. In this setting, the number of sketches $q = m$ for $\mathbf{A} \in \mathbb{R}^m$, and the sketch size is $\tau = 1$. In order to perform the adaptive update efficiently, the matrices

$$\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i \mathbf{C}_i = \frac{\mathbf{A}_{i:}^\top}{\|\mathbf{A}_{i:}\|} \quad \text{and} \quad \mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_j \mathbf{C}_j = \frac{\langle \mathbf{A}_{i:}, \mathbf{A}_{j:} \rangle}{\|\mathbf{A}_{i:}\| \|\mathbf{A}_{j:}\|} \; \forall\, i, j = 1, 2, \ldots m$$

should be precomputed.

In order to succinctly express the convergence rates, we define the diagonal probability matrix $\mathbf{P} = \text{Diag}\,(p_1, \ldots, p_m)$ and the normalized matrix $\bar{\mathbf{A}} \overset{\text{def}}{=} \mathbf{D}_{RK}^{-1}\mathbf{A}$, with $\mathbf{D}_{RK} \overset{\text{def}}{=} \text{Diag}\,(\|\mathbf{A}_{1:}\|_2, \ldots, \|\mathbf{A}_{m:}\|_2)$ as in [NSL16]. In the randomized Kaczmarz setting, the projection matrix $\mathbf{Z}_i$ as defined in Equation (2.16) is the orthogonal projection onto the i^{th} row of $\mathbf{A}$

and takes the form

$$\mathbf{Z}_i = \frac{\mathbf{A}_{i:}\mathbf{A}_{i:}^\top}{\|\mathbf{A}_{i:}^2\|}.$$

We then have

$$\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right] = \mathbf{D}_{RK}^{-1}\mathbf{A}\mathbf{P}\mathbf{A}^\top\mathbf{D}_{RK}^{-1} = \bar{\mathbf{A}}^\top\mathbf{P}\bar{\mathbf{A}}.$$

The costs and convergence rates for the adaptive sampling strategies discussed in Section 2.6 applied to the Kaczmarz method are summarized in Table 2.2, where we use the notation $\|x\|_\infty \overset{\text{def}}{=} \max_i |x_i|$ for any vector x.

Sampling strategy	Convergence rate bound	Shown in	Flops per iteration
Uniform	$1 - \frac{1}{m}\lambda_{\min}^+(\bar{\mathbf{A}}^\top\bar{\mathbf{A}})$	[NSL16], Theorem 2.7.4	$2\min(n,m) + 2n$
$p_i \propto \|\mathbf{A}_{i:}\|_2^2$	$1 - \frac{\lambda_{\min}^+(\mathbf{A}^\top\mathbf{A})}{\|\mathbf{A}\|_F^2}$	[SV09], Theorem 2.7.4	$2\min(n,m) + 2n$
Max-distance	$1 - \min_{v\in\text{Range}\left(\mathbf{A}^\top\right)} \frac{\|\bar{\mathbf{A}}v\|_\infty}{\|v\|_2}$	[NSL16], Theorem 2.7.5	$3m + 2n$
$p_i^k \propto f_i(x^k)$	$1 - \frac{m+1}{m}\lambda_{\min}^+(\bar{\mathbf{A}}^\top\bar{\mathbf{A}})$	Theorem 2.7.8	$5m + 2n$
Capped	$1 - (\theta\gamma + (1-\theta))\,\lambda_{\min}^+(\bar{\mathbf{A}}^\top\mathbf{P}\bar{\mathbf{A}})$	[BW18b], Theorem 2.7.10	$9m + 2n$

Table 2.2: Summary of convergence guarantees and costs of various sampling strategies for the randomized Kaczmarz method. Here, $\gamma = 1/\max_{i=1,\ldots,m}\sum_{j=1,j\neq i}^m p_i$ as defined in Equation (2.37), $\mathbf{P} = \text{Diag}\left(p_1,\ldots,p_m\right)$ is a matrix of arbitrary fixed probabilities, and $\bar{\mathbf{A}} := \mathbf{D}_{RK}^{-1}\mathbf{A}$, with $\mathbf{D}_{RK} := \text{Diag}\left(\|\mathbf{A}_{1:}\|_2,\ldots,\|\mathbf{A}_{m:}\|_2\right)$. Only leading-order flop counts are reported. The number of sketches is q, the sketch size is τ and the number of rows and columns in the matrix $\mathbf{A}$ are m and n respectively.

2.9.2 Adaptive coordinate descent

By choosing the parameter matrix $\mathbf{B} = \mathbf{A}^\top\mathbf{A}$ and sketching matrices $\mathbf{S}_i = \mathbf{A}e_i$ for $i = 1,\ldots,n$ where $e_i \in \mathbb{R}^m$ is the i^{th} coordinate vector, we arrive at the coordinate descent method discussed in Subsection 2.1.2. In this setting, the number of sketches $q = n$, where n is

number of columns in $\mathbf{A}$, and the sketch size is $\tau = 1$.

Coordinate descent uses fewer flops per iteration than indicated by the general computation that will be given in Subsection 2.A.1. This computational savings arises from the sparsity of the matrix $\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}\mathbf{C}_{i_k} = e_i/\|\mathbf{A}_{:i}\|$. As a result, the iterate update of x^k to x^{k+1} using the sketched residuals $\mathbf{R}_{i_k}^k$ requires only $O(1)$ flops instead of $2n$ flops as indicated in the general analysis that is summarized in Table 2.5. The cost of a coordinate descent update is dominated by the $2n$ flops required to calculate $\mathbf{R}_{i_k}^k$ by either the auxiliary update of Line 11 of Algorithm 5 or directly via Equation (2.53).

Similar to the randomized Kaczmarz case, we define the diagonal probability matrix $\mathbf{P} \overset{\text{def}}{=} \text{Diag}\,(p_1, \ldots, p_n)$ and the normalized matrix $\widetilde{\mathbf{A}} \overset{\text{def}}{=} \mathbf{A}\mathbf{D}_{CD}^{-1}$, with $\mathbf{D}_{CD} \overset{\text{def}}{=} \text{Diag}\,(\|\mathbf{A}_{:1}\|_2, \ldots, \|\mathbf{A}_{:n}\|_2)$. The projection matrix $\mathbf{Z}_i$ as defined in Equation (2.16) is the projection given by

$$\mathbf{Z}_i = (\mathbf{A}^\top\mathbf{A})^{-1/2}\mathbf{A}^\top\mathbf{A}\frac{e_ie_i^\top}{\|\mathbf{A}_{:i}\|^2}\mathbf{A}^\top\mathbf{A}(\mathbf{A}^\top\mathbf{A})^{-1/2} = (\mathbf{A}^\top\mathbf{A})^{1/2}\frac{e_ie_i^\top}{\|\mathbf{A}_{:i}\|^2}(\mathbf{A}^\top\mathbf{A})^{1/2}.$$

We then have

$$\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right] = (\mathbf{A}^\top\mathbf{A})^{1/2}\mathbf{D}_{CD}^{-1}\mathbf{P}\mathbf{D}_{CD}^{-1}(\mathbf{A}^\top\mathbf{A})^{1/2}.$$

Note that $\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]$ is similar to $\mathbf{P}\mathbf{D}_{CD}^{-1}\mathbf{A}^\top\mathbf{A}\mathbf{D}_{CD}^{-1} = \mathbf{P}\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}$ and thus

$$\lambda_{\min}^+(\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]) = \lambda_{\min}^+(\mathbf{P}\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}).$$

The costs and convergence rates for the adaptive sampling strategies discussed in Section 2.6 applied to coordinate descent are summarized in Table 2.3.

2.10 Experiments

We test the performance of various adaptive and non-adaptive sampling strategies in the special sketch-and-project subcases of randomized Kaczmarz and coordinate descent. We report performance via three different measurements: norm-squared error versus iteration, norm-squared error versus approximate flop count, and the worst expected convergence factor.

Sampling strategy	Convergence rate bound	Shown in	Flops per iteration
Uniform	$1 - \frac{1}{n}\lambda_{\min}^{+}(\widetilde{\mathbf{A}}^{\top}\widetilde{\mathbf{A}})$	Theorem 2.7.4	$2n$
$p_i \propto \|\mathbf{A}_{:i}\|_2^2$	$1 - \frac{\lambda_{\min}^{+}(\mathbf{A}^{\top}\mathbf{A})}{\|\mathbf{A}\|_F^2}$	[LL10] Theorem 2.7.4	$2n$
Max-distance	$1 - \min_{v \in \text{Range}\left(\mathbf{A}^{\top}\right)} \frac{\|\widetilde{\mathbf{A}}v\|_{\infty}}{\|v\|_2}$	Theorem 2.7.5	$3n$
$p_i^k \propto f_i(x^k)$	$1 - \frac{n+1}{n}\lambda_{\min}^{+}(\widetilde{\mathbf{A}}^{\top}\widetilde{\mathbf{A}})$	Theorem 2.7.8	$5n$
Capped	$1 - (\theta\gamma + (1-\theta))\,\lambda_{\min}^{+}(\mathbf{P}\widetilde{\mathbf{A}}^{\top}\widetilde{\mathbf{A}})$	Theorem 2.7.10	$9n$

Table 2.3: Summary of convergence guarantees and costs of various sampling strategies for adaptive coordinate descent. Here, $\gamma = 1/\max_{i=1,\ldots,m}\sum_{j=1,j\neq i}^{n} p_i$ as defined in Equation (2.37), $\mathbf{P} = \text{Diag}\,(p_1,\ldots,p_n)$ is a matrix of arbitrary fixed probabilities, and $\widetilde{\mathbf{A}} = \mathbf{A}\mathbf{D}_{CD}^{-1}$, with $\mathbf{D}_{CD} = \text{Diag}\,(\|\mathbf{A}_{:1}\|_2,\ldots,\|\mathbf{A}_{:n}\|_2)$. Only flop counts of leading-order are reported.

Results are averaged over 50 trials. For each trial a single matrix $\mathbf{A}$ is used. For the experiments measuring error, a single true solution $x^{\star}$ and vector b are used. The worst expected convergence factor aims to approximate the spectral constants of Definition 2.7.1. Since the max-distance method is deterministic, generating a new exact solution $x^{\star}$ adds variation between trials, hopefully leading to a more accurate approximation of the spectral constant. The exact solutions $x^{\star}$ are generated by

$$x^{\star} = \frac{\mathbf{A}^{\top}\omega}{\|\mathbf{A}^{\top}\omega\|_{\mathbf{B}}},$$

where $\omega \in \mathbb{R}^m$ is a vector of i.i.d. random normal entries. Thus $\|x^{\star}\|_{\mathbf{B}}^2 = 1$ is normalized with respect to the $\mathbf{B}$–norm and lies in the row space of $\mathbf{A}$. The latter condition guarantees that $x^{\star}$ is indeed the unique solution to Equation (2.1). We measure the error in terms of the $\mathbf{B}$-norm. Recall that for randomized Kaczmarz $\mathbf{B} = \mathbf{I}$, while for coordinate descent, $\mathbf{B} = \mathbf{A}^{\top}\mathbf{A}$. The sketch-and-project methods are implemented using the auxiliary update Line 11 of Algorithm 5. For the max-distance rule, if multiple sketches achieve the maximal sketched-loss value, we select the first such sketch.

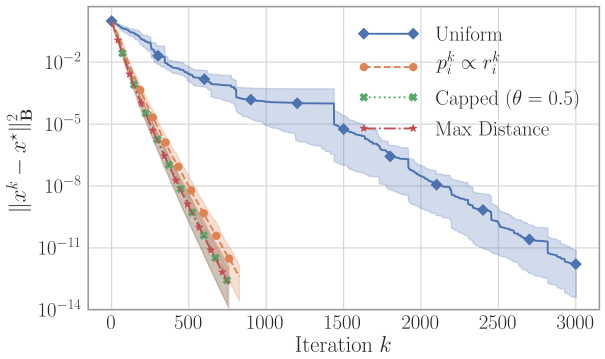
We consider synthetic matrices of size $1,000 \times 100$ and $100 \times 1,000$ that are generated with i.i.d. standard Gaussian entries. We additionally test the various adaptive sampling strategies on two large-scale matrices arising from real world problems. These matrices are available via the SuiteSparse Matrix Collection [DH11]. The first system (Ash958) is an overdetermined matrix with 958 rows, 292 columns, and 1,916 entries [DGL89, DGL92]. The matrix comes from a survey of the United Kingdom and is part of the original Harwell sparse matrix test collection [DGL92]. The second real matrix we consider is the GEMAT1 matrix, which arises from optimal power flow modeling. This matrix is highly underdetermined and consists of 4,929 rows, 10,595 columns, and 47,369 entries [DGL89, DGL92].

2.10.1 Error per iteration

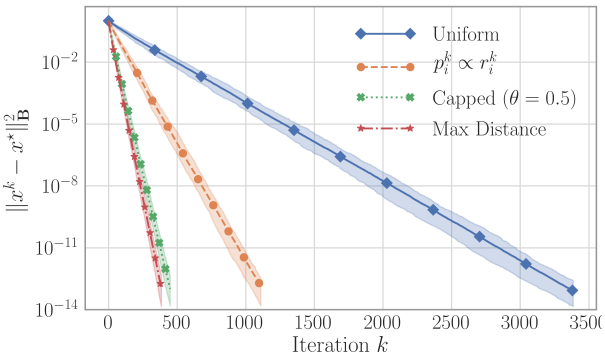
We first investigate the convergence of the squared norm of the error $\|x^k - x^\star\|_{\mathbf{B}}^2$ versus the number of iterations in Figure 2.2. The first row of subfigures (Figures 2.2a and 2.2b) shows convergence for randomized Kaczmarz, while the second row of subfigures (Figures 2.2c and 2.2d) gives the convergence of various sampling strategies for coordinate descent. The first column of subfigures (Figures 2.2a and 2.2c) uses an underdetermined system of $100 \times 1,000$ while the second column of subfigures (Figures 2.2b and 2.2d) considers an overdetermined system of $1,000 \times 100$. Figures 2.4c and 2.4d demonstrate convergence per iteration for the Ash958 matrix and Figures 2.5a and 2.5c for randomized Kaczmarz and coordinate descent applied to the GEMAT1 matrix.

As expected, we see that the max-distance rule performs the best per iteration followed by the capped adaptive strategy, then sampling proportional to the sketched residuals and finally followed by the uniform strategy. For randomized Kaczmarz applied to underdetermined systems and coordinate descent applied to overdetermined systems, max-distance and the capped adaptive sampling strategies perform similarly in terms of squared error per iteration. The convergence of randomized Kaczmarz for each sampling strategy applied to overdetermined systems is very similar to that of coordinate descent applied to underdetermined

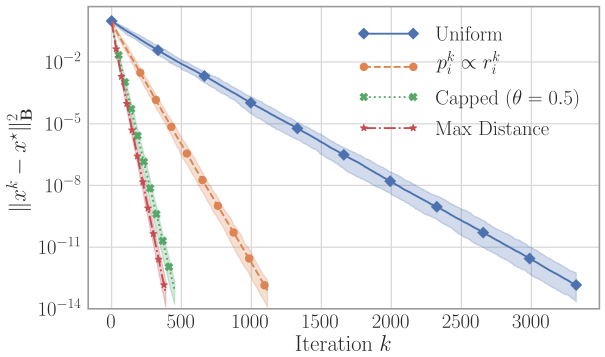
systems. Conversely, the convergence of randomized Kaczmarz for each sampling strategy applied to underdetermined systems is very similar to that of coordinate descent applied to overdetermined systems. For the large and underdetermined GEMAT1 matrix, we find that randomized coordinate descent methods have much larger variance in their performance compared to randomized Kaczmarz methods.



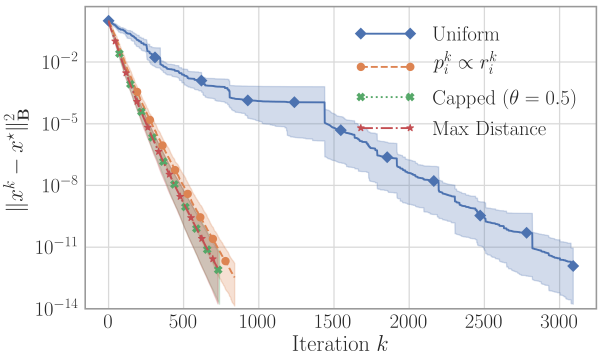
(a) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{100\times1,000}$.



(b) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{1,000\times100}$.



(c) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{100\times1,000}$.



(d) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{1,000\times100}$.

Figure 2.2: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent methods on matrices with i.i.d. standard Gaussian entries. Squared error norms were averaged over 50 trials. The shaded areas indicate the middle 95% performance. Subplots on the left show convergence for underdetermined systems, while those on the right show the convergence for overdetermined systems.

2.10.2 Error versus approximate flops required

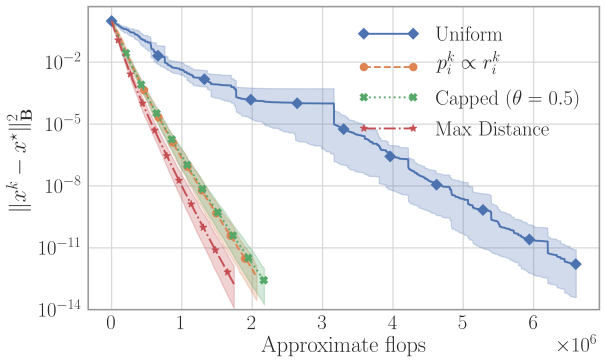
If we take into account the number of flops required for each method, the relative performance of the methods changes significantly. In order to approximate the number of flops required for each sampling strategy, we use the leading-order flop counts per iteration given in Tables 2.2 and 2.3. We do not consider the precomputational costs, but only the costs incurred at each iteration. The performance in terms of flops of each sampling strategy is reported in Figure 2.3. Performance on the Ash958 matrix is reported in Figures 2.4c and 2.4d. Performance on the GEMAT1 matrix for randomized Kaczmarz and coordinate descent is reported in Figures 2.5b and 2.5d.

As discussed in Section 2.8, the adaptive methods are typically more expensive than non-adaptive methods as one must update the sketched residuals $\mathbf{R}_i^k$ for $i = 1, \ldots, q$ at each iteration k. Yet even after taking flops into consideration, we find that the max-distance rule still performs the best overall. For randomized Kaczmarz applied to an overdetermined synthetic matrix, uniform sampling performance is comparable to max-distance (Figure 2.3b). In all other experiments, however, max-distance is the clear winner. Since the max-distance rule performs at least as well per iteration as the other adaptive methods, yet the max-distance rule is less expensive, it naturally outperforms the other adaptive methods when flop counts are considered.

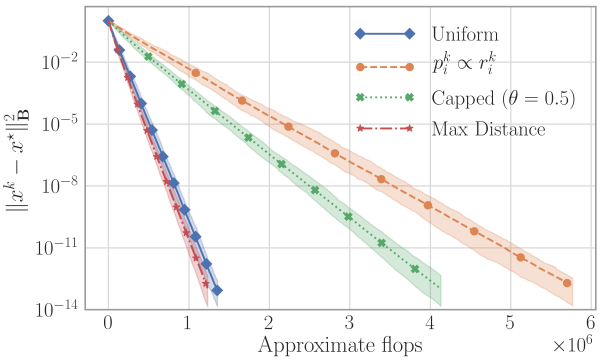
2.10.3 Spectral constant estimates

Theorems 2.7.4, 2.7.5 and 2.7.7 to 2.7.10 of Section 2.7 provide conservative views of the convergence rates of each method, as the spectral constants of Definition 2.7.1 give the expected convergence corresponding to the worst possible point $x \in \text{Range}\,(\mathbf{B}^{-1}\mathbf{A})$ as opposed to the iterates x^k. In practice, the convergence at each iteration performs better than the convergence bounds indicate.

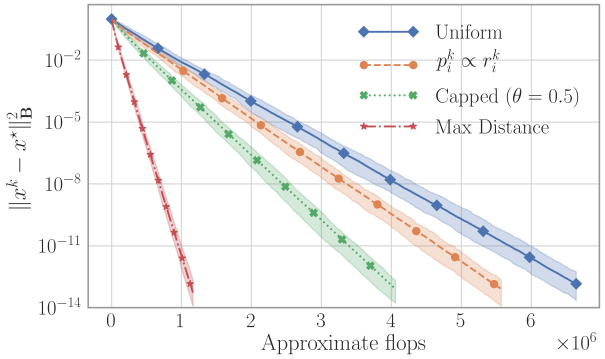
Recall that the convergence rates derived in Section 2.7 are given in terms of spectral



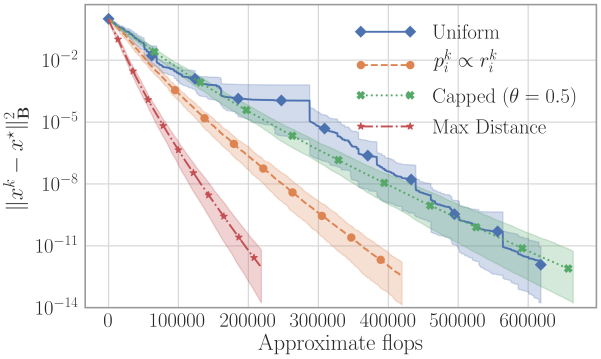
(a) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{100\times 1{,}000}$.



(b) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{1{,}000\times 100}$.



(c) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{100\times 1{,}000}$.



(d) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{1{,}000\times 100}$.

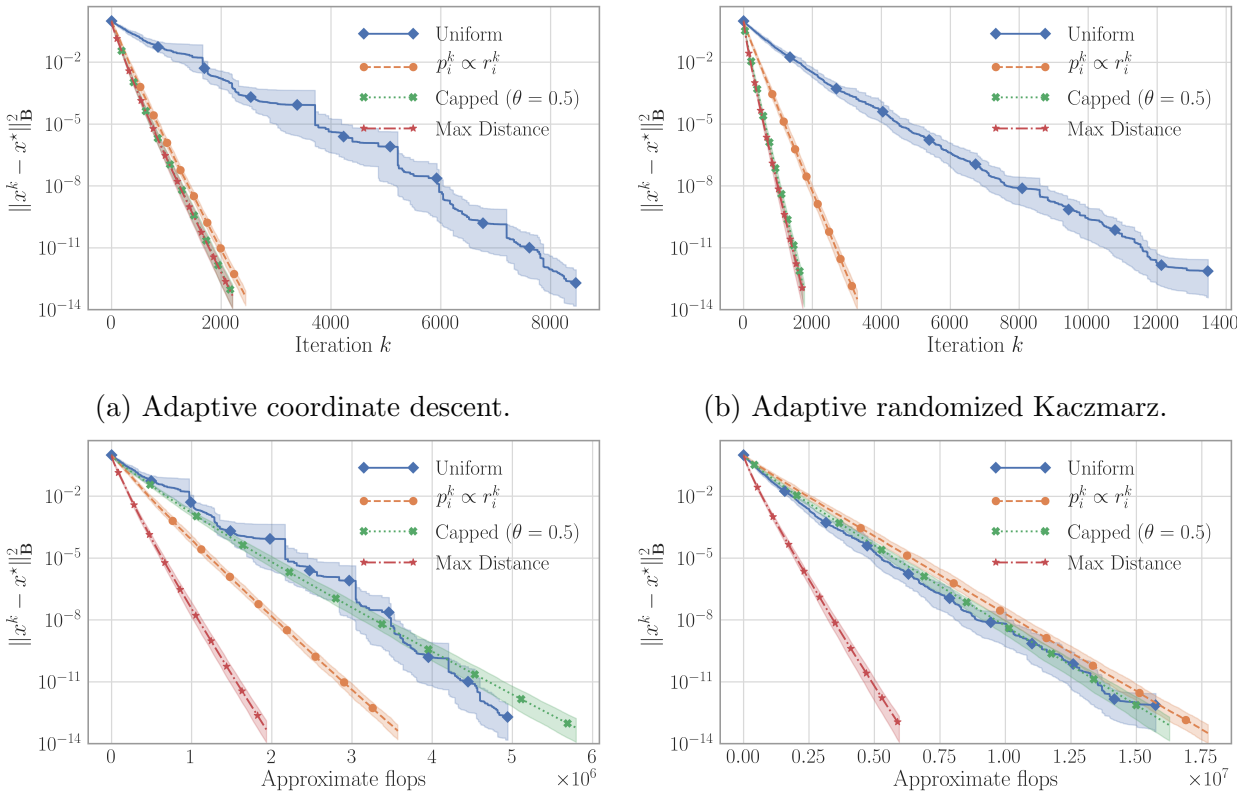
Figure 2.3: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent methods on matrices with i.i.d. standard Gaussian entries. Squared error norms were averaged over 50 trials and are plotted against the approximate flops aggregated over the computations that occur at each iteration. The shaded areas indicate the middle 95% performance. Subplots on the left show convergence for underdetermined systems, while those on the right show the convergence for overdetermined systems.

constants (Definition 2.7.1) of the form

$$\sigma_p^2(\mathbf{B}, \mathbf{S}) \stackrel{\text{def}}{=} \min_{x \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)} \frac{\mathbb{E}_{i\sim p}\left[f_i(x)\right]}{\|x - x^\star\|_{\mathbf{B}}^2}.$$

We will refer to the value

$$\frac{\mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right]}{\|x^k - x^\star\|_{\mathbf{B}}^2}$$



(a) Adaptive coordinate descent.

(b) Adaptive randomized Kaczmarz.

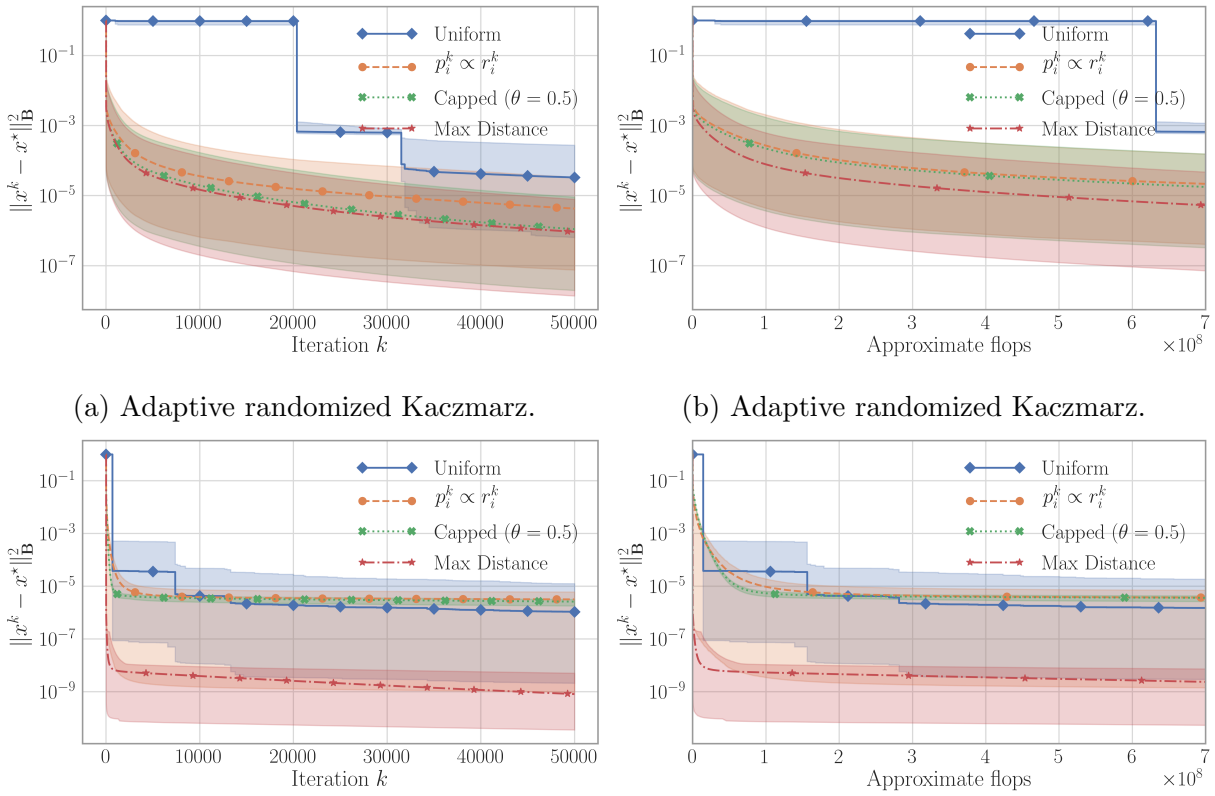
(c) Adaptive coordinate descent.

(d) Adaptive randomized kaczmarz.

Figure 2.4: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent methods on the Ash958 matrix. Squared error norms were averaged over 50 trials and plotted against both the iteration and the approximate flops required. The shaded areas indicate the middle 95% performance.

as the *expected step-size factor* and note that larger values indicate superior performance.

The smallest expected step-size factor observed for each method provides an estimate and upper bound on the spectral constants in the derived convergence rates. The minimal expected step-size factor for each sampling method applied to random Gaussian matrices of size $1,000 \times 100$ and $100 \times 1,000$ are reported in Table 2.4. As expected, we find that uniform sampling has the smallest value while max-distance has the largest. In Theorem 2.7.8, we proved a bound on the convergence rate for sampling proportional to the sketched losses that was strictly faster than the convergence guarantee for uniform sampling. We find that the



(a) Adaptive randomized Kaczmarz.

(b) Adaptive randomized Kaczmarz.

(c) Adaptive coordinate descent.

(d) Adaptive coordinate descent.

Figure 2.5: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent on the GEMAT1 matrix. Squared error norms were averaged over 50 trials and plotted against both the iteration and the approximate flops required. The shaded areas indicate the middle 95% performance.

estimated spectral constants in Table 2.4 for the proportional sampling strategy are likewise strictly larger than the estimated spectral constants for uniform sampling.

2.11 Conclusions

We extended adaptive sampling methods to the general sketch-and-project setting. We presented a computationally efficient method for implementing the adaptive sampling strategies using an auxiliary update. For several specific adaptive sampling strategies including

Sampling strategy	Randomized Kaczmarz		Coordinate descent	
	$1,000 \times 100$	$100 \times 1,000$	$1,000 \times 100$	$100 \times 1,000$
Uniform	0.00705	0.00667	0.00656	0.00715
$p_i \propto \|\mathbf{A}_{:i}\|_2^2$	0.02019	0.01569	0.01722	0.02014
Capped	0.03885	0.01901	0.01952	0.03878
Max-distance	0.04593	0.01994	0.02171	0.04711

Table 2.4: Minimal expected step-size factor for each sampling method applied to matrices of i.i.d. Gaussian entries.

max-distance selection, the capped adaptive sampling of [BW18a, BW18b, BW19a], and sampling proportional to the sketched residuals, we derived convergence rates which show that the max-distance rule has the fastest convergence guarantee among the sampling methods considered. This superior performance was observed in practice as well for both the randomized Kaczmarz and coordinate descent subcases.

2.A Implementation tricks and computational complexity, cont.

We describe how one can perform adaptive sketching with the same order of cost per iteration as the standard non-adaptive sketch-and-project method when τq, the number of sketches q times the sketch size τ, is not significantly larger than the number of columns n. In particular, we show how adaptive sketching methods can be performed for a per-iteration cost of $O(\tau^2 q + \tau n)$, whereas the standard non-adaptive sketch-and-project method has a per-iteration cost of $O(\tau n)$. The precomputations and efficient update strategies presented here are a generalization of those suggested in [BW18a] for the Kaczmarz setting. Precomputational costs are a one time expense and are independent of the sampling strategy. The precomputational costs depend on the sparsity structure of the sketches and are summarized for randomized Kaczmarz and coordinate descent in Table 2.6. The computational costs given in this section

may be over-estimates of the costs required for specific sketch choices such as when the update is sparse, as is the case in coordinate descent. The special cases of adaptive Kaczmarz and adaptive coordinate descent were analyzed in Section 2.9.

Pseudocode for efficient implementation is provided in Algorithm 5. Throughout this section, we will frequently omit $O(1)$ and $O(\log(q))$ flop counts since they are insignificant compared to the number of rows m, the number of columns n, and the number of sketches q.

2.A.1 Per-iteration cost

The main computational costs of adaptive sketch-and-project (Algorithm 2) at each iteration come from computing the sketched losses $f_i(x^k)$ of Equation (2.8) and updating the iterate from x^k to x^{k+1} via Equation (2.6). We now discuss how these steps can be calculated efficiently. An efficient implementation for adaptive sketch-and-project is provided in Algorithm 5. The costs of each step of an iteration of the adaptive sketch-and-project method are summarized in Table 2.5.

Let $\mathbf{C}_i$ be any square matrix satisfying

$$\mathbf{C}_i\mathbf{C}_i^\top = (\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger. \tag{2.54}$$

For example, $\mathbf{C}_i$ could be the Cholesky decomposition of $(\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger$. The sketched loss $f_i(x^k)$ and the iterate update from x^k to x^{k+1} can now be written as

$$f_i(x^k) = \|\mathbf{S}_i^\top(\mathbf{A}x^k - b)\|^2_{\mathbf{C}_i\mathbf{C}_i^\top} = \|\mathbf{C}_i^\top \mathbf{S}_i^\top(\mathbf{A}x^k - b)\|_2^2$$

and

$$x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k}\mathbf{C}_{i_k}\mathbf{C}_{i_k}^\top \mathbf{S}_{i_k}^\top(\mathbf{A}x^k - b).$$

Notice that both the iterate update for x^k and the formula for the sketched loss $f_i(x^k)$ share the sketched residual $\mathbf{R}_i^k \overset{\text{def}}{=} \mathbf{C}_i^\top \mathbf{S}_i^\top(\mathbf{A}x^k - b)$ defined in Equation (2.53). In adaptive methods one must compute the sketched residual $\mathbf{R}_i^k$ for $i = 1, 2, \ldots, q$. When sampling from a fixed distribution, however, calculating the sketched losses $f_i(x^k)$ is unnecessary and only the sketched residual $\mathbf{R}_{i_k}^k$ corresponding to the selected index i_k need be computed.

Depending on the sketching matrices $\mathbf{S}_i$ and the matrix $\mathbf{B}$, it is possible to update the iterate x^k and compute the sketched losses $f_i(x^k)$ more efficiently if one maintains the set of sketched residuals $\{\mathbf{R}_i^k : i = 1, 2, \ldots, q\}$ in memory. Using the sketched residuals, the calculations above can be rewritten as

$$f_i(x^k) = \|\mathbf{R}_i^k\|_2^2 \tag{2.55}$$

and

$$x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k} \mathbf{C}_{i_k} \mathbf{R}_{i_k}^k. \tag{2.56}$$

The sketched residuals $\{\mathbf{R}_i^k : i = 1, 2, \ldots, q\}$ can either be computed via an auxiliary update applied to the set of previous set of sketched residuals $\{\mathbf{R}_i^{k-1} : i = 1, 2, \ldots, q\}$ or directly using the iterate x^k. Using the auxiliary update,

$$\begin{aligned}
\mathbf{R}_i^{k+1} &= \mathbf{C}_i^\top \mathbf{S}_i^\top (\mathbf{A}x^{k+1} - b) \\
&= \mathbf{C}_i^\top \mathbf{S}_i^\top \Big(\mathbf{A}(x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k} \mathbf{C}_{i_k} \mathbf{R}_{i_k}^k) - b \Big) \\
&= \mathbf{R}_i^k - \mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k} \mathbf{C}_{i_k} \mathbf{R}_{i_k}^k
\end{aligned} \tag{2.57}$$

with the initialization

$$\mathbf{R}_i^0 = \mathbf{C}_i^\top \left(\mathbf{S}_i^\top (\mathbf{A}x^0 - b) \right).$$

If the matrix $\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_j \mathbf{C}_j \in \mathbb{R}^{\tau \times \tau}$ is precomputed for each $i, j = 1, 2, \ldots, q$, the sketched residual $\mathbf{R}_i^k$ can be updated to $\mathbf{R}_i^{k+1}$ for $2\tau^2$ flops for each index i via Equation (2.57). Using the precomputed matrices requires storing $\frac{1}{4}\tau(\tau+1)q(q+1)$ floats.

In the non-adaptive case, one only needs to compute the single sketched residual $\mathbf{R}_{i_k}^k$ as opposed to the entire set of sketched residuals, since the sketched losses $f_i(x^k)$ are not needed. If the matrices

$$\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A} \in \mathbb{R}^{\tau \times n} \quad \text{and} \qquad \mathbf{C}_i^\top \mathbf{S}_i^\top b \in \mathbb{R}^\tau,$$

are precomputed for $i = 1, 2, \ldots, q$, computing each sketched residual $\mathbf{R}_i^k$ directly from the iterate x^k costs $2\tau n$ flops via Equation (2.53). If $q\tau > n$, then it is cheaper to compute the

sketched residual $\mathbf{R}_{i_k}^k$ using the auxiliary update Equation (2.57) rather than computing it directly from x^k.

From the sketched residual $\mathbf{R}_i^k$, the sketched losses $f_i(x^k)$ can be computed for $2\tau - 1$ flops for each index i via Equation (2.55). If the matrix $\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i\mathbf{C}_i \in \mathbb{R}^{n\times\tau}$ is precomputed for each $i = 1, 2, \ldots, q$, the iterate x^k can then be updated to x^{k+1} for $2\tau n$ flops via Equation (2.56). These costs are summarized in Table 2.5.

Algorithm 5 Efficient Adaptive Sampling Sketch-and-Project

1: **input:** $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\{\mathbf{S}_i \in \mathbb{R}^{m\times\tau} : i = 1, 2, \ldots, q\}$, $\mathbf{B} \in \mathbf{R}^{n\times n}$, $x^0 \in \text{Range}\left(\mathbf{B}^{-1}\mathbf{A}^\top\right)$,
2: compute $\mathbf{C}_i = \text{Cholesky}\left((\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i)^\dagger\right)$ for $i = 1, 2, \ldots, q$
3: ▷ The $\mathbf{C}_i$ can be discarded after Line 6.
4: compute $\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i\mathbf{C}_i \in \mathbb{R}^{n\times\tau}$ for $i = 1, 2, \ldots, q$
5: compute $\mathbf{C}_i^\top\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_j\mathbf{C}_j \in \mathbb{R}^{\tau\times\tau}$ for $i, j = 1, 2, \ldots, q$
6: initialize $\mathbf{R}_i^0 = \mathbf{C}_i^\top\left(\mathbf{S}_i^\top(\mathbf{A}x^0 - b)\right) \in \mathbb{R}^\tau$ for $i = 1, 2, \ldots, q$
7: **for** $k = 0, 1, 2, \ldots$
8: compute $f_i(x^k) = \|\mathbf{R}_i^k\|_2^2$ for $i = 1, 2, \ldots, q$
9: sample $i_k \sim p_i^k$, where $p^k \in \Delta_q$ is a function of $f(x^k)$
10: update $x^{k+1} = x^k - (\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}\mathbf{C}_{i_k})\mathbf{R}_{i_k}^k$
11: update $\mathbf{R}_i^{k+1} = \mathbf{R}_i^k - (\mathbf{C}_i^\top\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}\mathbf{C}_{i_k})\mathbf{R}_{i_k}^k$ for $i = 1, 2, \ldots, q$
12: **output:** last iterate x^{k+1}

2.A.2 Cost of sampling indices

The cost of computing the sampling probabilities p^k from the sketched losses $f_i(x^k)$ depends on the sampling strategy used. Sampling from a fixed distribution can be achieved with an $O(1)$ cost using precomputations of $O(q)$ [Wal74]. Adaptive strategies sample from a new, unseen distribution at each iteration, which can be achieved with a mean of q flops using, for example, inversion by sequential search [Kem81, Dev86, p. 86]. In practice, the

Per iteration computation	Flops
$f_i(x^k)\ \forall i$ via Eqn. (2.55)	$(2\tau - 1)q$
x^{k+1} via Eqn. (2.56)	$2\tau n$
$\mathbf{R}_i^k\ \forall i$ via Eqn. (2.57)	$2\tau^2 q$
$\mathbf{R}_{i_k}^k$ via Eqn. (2.53)	$2\tau n$

(a) Baseline flop counts. Flop counts of $O(1)$ have been omitted.

Stored object	Storage
x^k	n
$\mathbf{R}_i^k \quad \forall i$	τq
$\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i\mathbf{C}_i \quad \forall i$	$\tau q n$
$\mathbf{C}_i^\top\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_j\mathbf{C}_j \forall i,j$	$\frac{1}{4}\tau(\tau+1)q(q+1)$
$\mathbf{C}_i^\top\mathbf{S}_i^\top\mathbf{A}$ and $\mathbf{C}_i^\top\mathbf{S}_i^\top b\ \forall i$	$\tau q(n+1)$

(b) Storage costs.

Table 2.5: Summary of the costs of Algorithm 5 excluding costs that are specific to the sampling method. The number of sketches is q, the sketch size is τ and the number of columns in the matrix $\mathbf{A}$ is n.

probabilities p_i^k corresponding to each index i are given by a function of the sketched losses $f(x_i^k)$ and normalizing these values is unnecessary. Instead, one can sum the q sketched losses and apply inversion by sequential search with a uniform random value r generated between zero and the sum of these values. This summation requires $q-1$ flops. Thus, the total cost for sampling from an adaptive probability distribution for the methods considered is approximately $2q$ flops on average. The costs for the sampling strategies discussed in Section 2.6 are summarized in Tables 2.7 and 2.8. The calculations of these costs are discussed in more detail in Subsection 2.A.3.

2.A.3 Sampling strategy specific costs

We now detail the calculations that lead to the costs associated with each of the specific sampling strategies that are reported in Tables 2.7 and 2.8.

Abstract computation	Randomized Kaczmarz		Coordinate Descent	
	Computation	Cost	Computation	Cost
$\mathbf{C}_i$ of Equation (2.54)	$\frac{1}{\|\mathbf{A}_{i:}\|}$	$2mn + O(m)$	$\frac{1}{\|\mathbf{A}_{:i}\|}$	$2mn + O(n)$
$\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i\mathbf{C}_i$	$\frac{\mathbf{A}_{i:}^\top}{\|\mathbf{A}_{i:}\|}$	mn	$\frac{e_i}{\|\mathbf{A}_{:i}\|}$	n
$\mathbf{C}_i^\top\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_j\mathbf{C}_j$	$\frac{\langle\mathbf{A}_{i:},\mathbf{A}_{j:}\rangle}{\|\mathbf{A}_{i:}\|\|\mathbf{A}_{j:}\|}$	$m^2n + O(m^2 + mn)$	$\frac{\langle\mathbf{A}_{:i},\mathbf{A}_{:j}\rangle}{\|\mathbf{A}_{:i}\|\|\mathbf{A}_{:j}\|}$	$mn^2 + O(mn + n^2)$

Table 2.6: Precomputational costs for adaptive randomized Kaczmarz and adaptive coordinate descent. The computational costs assume the previous elements have been computed and give the cost of computing the value for all indices.

2.A.3.1 Sampling from a fixed distribution

When sampling the indices i from a fixed distribution, computing the sketched losses $f_i(x^k)$ is unnecessary and only the sketched residual $\mathbf{R}_{i_k}^k$ of the selected index i_k is needed to update the iterate x^k. If $q\tau > n$, where q is the number of sketches, τ is the sketch size and n is the number of columns in the matrix $\mathbf{A}$, it is cheaper to compute the sketched residual $\mathbf{R}_{i_k}^k$ using the auxiliary update Equation (2.57) rather than computing it directly from x^k. Ignoring the $O(1)$ cost of sampling from the fixed distribution, the iterate update takes either $4\tau n$ flops if $q\tau > n$ and one maintains the set of sketched residuals via the auxiliary update Equation (2.57) or $2\tau(n+q)$ flops if the sketched residual $\mathbf{R}_{i_k}^k$ is calculated from the iterate x^k directly via Equation (2.53).

2.A.3.2 Max-distance selection

Performing max-distance selection requires finding the maximum element of the length q vector of sketched losses given in Equation (2.55). Assuming the maximal element is uniformly distributed among the entries, the mean cost is $q + O(\log q)$ flops, where q flops are used to check each element and $O(\log q)$ flops arise from updates to the running maximal value. For convenience, we ignore the $O(\log q)$ flops and consider the cost of the selection step using the

Sampling strategy	Flops from sampling		Non-sampling flops
	$\tau > 1$	$\tau = 1$	
Fixed probabilities $p_i^k \equiv p_i$	$O(1)$	$O(1)$	$2\tau \min(n, \tau q) + 2\tau n$
Max-distance	q	$O(\log(q))$	$(2\tau^2 + 2\tau - 1)q + 2\tau n$
$p_i^k \propto f_i(x^k)$	$2q$	$2q$	$(2\tau^2 + 2\tau - 1)q + 2\tau n$
Capped	$6q$	$6q$	$(2\tau^2 + 2\tau - 1)q + 2\tau n$

Table 2.7: Rule-specific per-iteration costs of Algorithm 5. Only leading-order flop counts are reported. The non-sampling flops are those that are independent of the specific adaptive sampling method used and are those that correspond to the steps indicated in Table 2.5a. The extra flops for sampling are those that are required to calculate the adaptive sampling probabilities p^k at each iteration. The number of sketches is q, the sketch size is τ and the number of columns in the matrix $\mathbf{A}$ is n.

max-distance rule to be q flops. If the sketches $\mathbf{S}_i$ are vectors, or equivalently we have $\tau = 1$, then the sketched residuals $\mathbf{R}_i^k$ are scalars and finding the maximal sketched loss $f_i(x^k)$ is equivalent to finding the sketched residual $\mathbf{R}_i^k$ of maximal magnitude. We can thus save q flops per iteration by skipping the step of computing the sketched losses and instead taking the sketched residual of maximal magnitude.

2.A.3.3 Sampling proportional to the sketched loss

Sampling indices with probabilities proportional to the sketched losses $f_i(x^k)$ requires approximately $2q$ flops on average using inversion by sequential search.

Sampling strategy	Flops per iteration	
	$\tau > 1$	$\tau = 1$
Fixed probabilities $p_i^k \equiv p_i$	$2\tau \min(n, \tau q) + 2\tau n$	$2\min(n, q) + 2n$
Max-distance	$(2\tau^2 + 2\tau)q + 2\tau n$	$3q + 2n$
$p_i^k \propto f_i(x^k)$	$(2\tau^2 + 2\tau + 1)q + 2\tau n$	$5q + 2n$
Capped	$(2\tau^2 + 2\tau + 5)q + 2\tau n$	$9q + 2n$

Table 2.8: Rule-specific per-iteration costs of Algorithm 5. Only leading-order flop counts are reported. The number of sketches is q, the sketch size is τ and the number of columns in the matrix $\mathbf{A}$ is n.

2.A.3.4 Capped adaptive sampling

Recall from Subsection 2.7.5 that using capped adaptive sampling requires identifying the set

$$\mathcal{W}_k = \left\{ i \mid f_i(x^k) \geq \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right] \right\}.$$

Sampling with the capped adaptive sampling strategy requires identifying the set $\mathcal{W}_k$ and sampling an index from this set. Identifying the set $\mathcal{W}_k$ requires $q + O(\log q)$ flops to identify the maximal sketched loss $f_i(x^k)$, $2q$ flops to compute the weighted average of the sketched losses $\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]$, $O(1)$ flops to calculate the threshold for the set $\mathcal{W}_k$, and q flops to compare each sketched loss against the threshold. Sampling an index from the set $\mathcal{W}_k$ requires on average $2q$ flops by using inversion by sequential search as discussed in Subsection 2.A.2.[§] Thus, the total cost of the sampling step is $6q + O(\log q)$ flops. When a uniform average is used in place of the weighted average, the expected sketched loss $\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]$ can be computed in just q flops as opposed to $2q$. In that case, the total cost of the sampling step is only $5q + O(\log q)$.

[§]The analyses of [BW18a, BW18b] omitted the cost of sampling the index from a new distribution at each iteration, and thus our cost calculations differ by $2q$.

2.B Auxiliary lemma

We now invoke a lemma taken from [GR16].

Lemma 2.B.1. *For any matrix* $\mathbf{W}$ *and symmetric positive semidefinite matrix* $\mathbf{G}$ *such that*

$$\operatorname{Null}(\mathbf{G}) \subset \operatorname{Null}\left(\mathbf{W}^{\top}\right), \tag{2.58}$$

we have that

$$\operatorname{Null}(\mathbf{W}) = \operatorname{Null}\left(\mathbf{W}^{\top}\mathbf{G}\mathbf{W}\right). \tag{2.59}$$

Proof. In order to establish Equation (2.59), it suffices to show the inclusion $\operatorname{Null}(\mathbf{W}) \supseteq \operatorname{Null}\left(\mathbf{W}^{\top}\mathbf{G}\mathbf{W}\right)$ since the reverse inclusion trivially holds. Letting $s \in \operatorname{Null}\left(\mathbf{W}^{\top}\mathbf{G}\mathbf{W}\right)$, we see that $\|\mathbf{G}^{1/2}\mathbf{W}s\|^2 = 0$, which implies $\mathbf{G}^{1/2}\mathbf{W}s = 0$. Consequently

$$\mathbf{W}s \in \operatorname{Null}\left(\mathbf{G}^{1/2}\right) = \operatorname{Null}(\mathbf{G}) \overset{(2.58)}{\subset} \operatorname{Null}\left(\mathbf{W}^{\top}\right).$$

Thus $\mathbf{W}s \in \operatorname{Null}\left(\mathbf{W}^{\top}\right) \cap \operatorname{Range}(\mathbf{W})$ which are orthogonal complements which shows that $\mathbf{W}s = 0$. □

CHAPTER 3

Randomized Kaczmarz with Averaging*

3.1 Introduction

In computed tomography, image processing, machine learning, and many other fields, a common problem is that of finding solutions to large linear systems of equations that do not fit in memory. Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, we aim to find $x \in \mathbb{R}^n$ which solves the linear system of equations

$$\mathbf{A}x = b. \tag{3.1}$$

We will generally assume the system is overdetermined, with $m \gg n$. For simplicity, we assume throughout that $\mathbf{A}$ has full rank so that the solution is unique when it exists. However, this assumption can be relaxed if one is interested in the least-norm there are multiple solutions.

When a solution to Equation (3.1) exists, we denote the solution by $x^\star$ and refer to the problem as *consistent*. Otherwise, the problem is *inconsistent*, and $x^\star$ instead denotes the *least-squares* solution

$$x^\star \stackrel{\text{def}}{=} \arg\min_{x \in \mathbb{R}^n} \frac{1}{2}\|b - \mathbf{A}x\|_2^2.$$

The least-squares solution can be equivalently written as $x^\star = \mathbf{A}^\dagger b$, where $\mathbf{A}^\dagger$ is the Moore–Penrose pseudoinverse of $\mathbf{A}$. We denote the least-squares *residual* as $r^\star \stackrel{\text{def}}{=} b - \mathbf{A}x^\star$, which is zero for consistent systems.

*This chapter is adapted from [MTM21].

3.1.1 Randomized Kaczmarz

Randomized Kaczmarz (RK) is a popular iterative method for approximating the least-squares solution of large, overdetermined linear systems [Kac37, SV09]. At each iteration, an equation is chosen at random from the system in Equation (3.1) and the current iterate is projected onto the solution space of that equation. The distribution for sampling the equation is a parameter of the method. In a relaxed variant of RK, a step is taken in the direction of this projection with the size of the step depending on a relaxation parameter.

Let x^k be the k^{th} iterate. We use $\mathbf{A}_i$ to denote the i^{th} row of $\mathbf{A}$ and $\|\cdot\| \stackrel{\text{def}}{=} \|\cdot\|_2$. The *relaxed RK* update is given by

$$x^{k+1} = x^k - \lambda_{k,i_k} \frac{\mathbf{A}_{i_k} x^k - b_{i_k}}{\|\mathbf{A}_{i_k}\|^2} \mathbf{A}_{i_k}^\top, \tag{3.2}$$

where i_k is sampled from some fixed distribution $\mathcal{D}$ at each iteration and $\lambda_{k,i}$ are relaxation parameters [CZT12]. Fixing the relaxation parameters $\lambda_{k,i} = 1$ for all iterations k and indices i leads to the standard RK method in which one projects the current iterate x^k onto the solution space of the chosen equation $\mathbf{A}_{i_k} x = b_{i_k}$ at each iteration [SV09]. Choosing relaxation parameters $\lambda_{k,i} \neq 1$ can be used to accelerate convergence or dampen the effect of noise in the linear system $\mathbf{A}x = b$ [CZT12, HN90a, HN90b].

For consistent systems, RK converges exponentially in expectation to the solution $x^\star$ [SV09], which when multiple solutions exist is the least-norm solution [ZF13, MNR15]. For inconsistent systems, there exists at least one equation $\mathbf{A}_j x = b_j$ that is not satisfied by $x^\star$. As a result RK cannot converge for inconsistent systems, since it will occasionally project onto the solution space of such an equation. One can, however, guarantee exponential convergence in expectation to within a radius of the least-squares solution [Nee10, ZF13, NT14]. This radius is commonly referred to as the *convergence horizon*. The size of the convergence horizon depends on several factors including the least squares residual $r^\star$, spectral properties of $\mathbf{A}$, and the choice of relaxation parameter λ_{k,i_k}.

3.1.2 Randomized Kaczmarz with Averaging

In order to take advantage of parallel computation and speed up the convergence of RK, we consider a simple extension of the RK method, where at each iteration multiple independent updates are computed in parallel and a weighted average of the updates is used. Specifically, we write the averaged RK update

$$x^{k+1} = x^k - \frac{1}{q}\sum_{i\in\tau_k} w_i \frac{\mathbf{A}_i x^k - b_i}{\|\mathbf{A}_i\|^2}\mathbf{A}_i^\top, \tag{3.3}$$

where τ_k is a random set of q row indices sampled *with replacement* and w_i represents the weight corresponding to the i^{th} row. RK with averaging is detailed in Algorithm 6. If τ_k is a set of size one and the weights are chosen as $w_i = 1$ for $i = 1, \ldots, m$, we recover the standard RK method.

Algorithm 6 Randomized Kaczmarz with Averaging

1: **Input** $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $x^0 \in \mathbb{R}^n$, weights $w \in \mathbb{R}^m$, number of maximum number of iterations K, distribution $\mathcal{D}$, number of threads q

2: **for** $k = 0, \ldots, K-1$

3: $\quad\tau_k \leftarrow q$ indices sampled from $\mathcal{D}$

4: $\quad$Compute $\delta \leftarrow \frac{1}{q}\sum_{i\in\tau_k} w_i \frac{\mathbf{A}_i x^k - b_i}{\|\mathbf{A}_i\|^2}\mathbf{A}_i^\top$ in parallel

5: $\quad$Update $x^{k+1} \leftarrow x^k - \delta$

6: **Output** x^K

3.1.3 Contributions

We derive a general convergence result for RK with averaging, and identify the conditions required for convergence to the least-squares solution. These conditions guide the choices of weights and probabilities of row selection, up to a relaxation parameter α. When $q = 1$ and appropriate weights and probabilities are chosen, we recover the standard convergence for RK [SV09, Nee10, ZF13].

For uniform weights and consistent systems, we relate RK with averaging to a more general parallel sketch-and-project method [RT20]. We also provide an estimate of the optimal choice for the relaxation parameter α, and compare to the estimated optimal relaxation parameter for the sketch-and-project method [RT20]. Through experiments, we show that our estimate lies closer to the empirically optimal value than the previously best known estimate.

3.1.4 Organization

In Section 3.2, we analyze the convergence of RK with averaging, and state our general convergence result in Subsection 3.2.2. In Section 3.3, we discuss the special case where the weights are chosen to be uniform, and in Section 3.4, we discuss the special case where the system is consistent. In Section 3.5, we derive an estimate of the optimal relaxation parameter for consistent systems. In Section 3.6, we experimentally explore the effects of the number of threads q, the relaxation parameter α, the weights w_i, and the distribution $\mathcal{D}$ on the convergence properties of RK with averaging.

3.1.5 Related Work

The Kaczmarz algorithm was originally proposed by Kaczmarz in 1937 [Kac37], though it was later independently developed by researchers in computed tomography as the algebraic reconstruction technique [GBH70, Byr07]. The original Kaczmarz method cycles through rows in a fixed order; however, this is known to perform poorly for certain orders of the rows [HS78]. Other Kaczmarz variants [XZ02] use deterministic methods to choose the rows, but their analysis is complicated and convergence results are overly pessimistic.

Some randomized control methods were proposed [HM93], but with no explicit proofs of convergence until Strohmer and Vershynin's 2009 paper [SV09], which proved RK converges linearly in expectation, with a rate directly related to spectral properties of the matrix A. This proof was later extended to inconsistent systems [Nee10, ZF13], showing convergence within a convergence horizon $\|r^\star\|/\sigma_{\min}(\mathbf{A})$ of the least-squares solution.

RK is a well-studied method with many variants. We do not provide an exhaustive review of the related literature [LL10, ZF13, NW13, CP12, EN11], but instead only remark on some closely related parallel extensions of RK.

Block Kaczmarz [Elf80, EHL81, AC89, NT14, XY15] randomly selects a block of rows from $\mathbf{A}$ at each iteration and computes its Moore–Penrose pseudoinverse. The pseudoinverse is then applied to the relevant portion of the current residual and added to the estimate, solving the least-squares problem only on the selected block of rows. Computing the pseudoinverse, however, is costly and difficult to parallelize.

The component-averaged row projections algorithm (CARP) [GG05] also distributes rows of $\mathbf{A}$ into blocks. However, instead of taking the pseudoinverse, the Kaczmarz method is then applied to the rows contained within each block. Multiple blocks are computed in parallel, and a component-averaging operator combines the approximations from each block. While CARP is shown to converge for consistent systems, and to converge cyclically for inconsistent systems, no exponential convergence rate is given.

AsyRK [LWS14] is an asynchronous parallel RK method that results from applying Hogwild! [NRR11] to the least-squares objective. In AsyRK, each thread chooses a row $\mathbf{A}_i$ at random and updates a random coordinate within the support of that row $\mathbf{A}_i$ with a weighted RK update. AsyRK is shown to have exponential convergence, given conditions on the step size. The analysis AsyRK requires that $\mathbf{A}$ is sparse, while we do not make this restriction.

RK falls under a more general class of methods often called sketch-and-project methods [GR15b]. For a linear system $\mathbf{A}x = b$, sketch-and-project methods iteratively project the current iterate onto the solution space of a sketched subsystem $\mathbf{S}^\top \mathbf{A}x - \mathbf{S}^\top b$. In particular, RK is a sketch-and-project method with $\mathbf{S}^\top = \mathbf{I}_i$, where $\mathbf{I}_i$ is the i^{th} row of the identity matrix. Other popular iterative methods such as coordinate descent can also be framed as sketch-and-project methods. In [RT20], the authors discuss a more general version of Algorithm 6 for sketch-and-project methods with averaging. Their analysis and discussion, however, focus on consistent systems and require uniform weights. We instead restrict our

analysis to RK, but allow inconsistent systems and general weights w_i.

RK can also be interpreted as a subcase of stochastic gradient descent (SGD) [RM51] applied to the loss function [NSW16]

$$F(x) = \sum_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} \frac{1}{2}(\mathbf{A}_i x - b_i)^2.$$

In this context, RK with averaging can be seen as mini-batch stochastic gradient descent [Bot98, NW17] with importance sampling, with the update

$$x^{k+1} = x^k - \frac{1}{q} \sum_{i \in \tau_k} \frac{w_i}{L_i} \nabla f_i(x),$$

where $L_i = \|\mathbf{A}_i\|^2$ is the Lipschitz constant of $\nabla f_i(x) = (\mathbf{A}_i x - b_i)\mathbf{A}_i^\top$.

3.2 Convergence of RK with Averaging

For inconsistent systems, RK satisfies the error bound

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] \leq \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}\right) \mathbb{E}\left[\|e^k\|^2\right] + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2}, \tag{3.4}$$

where $e^k \stackrel{\text{def}}{=} x^k - x^\star$ is the error of the k^{th} iterate, $\sigma_{\min}(\mathbf{A})$ is the smallest nonzero singular value of $\mathbf{A}$, the squared Frobenius-norm is $\|\mathbf{A}\|_F^2 = \sum_{i,j} \mathbf{A}_{ij}^2$ and $r^\star$ is the least-squares residual [Nee10, ZF13]. Iterating this error bound yields

$$\mathbb{E}\left[\|e^k\|^2\right] \leq \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}\right)^k \|e^0\|^2 + \frac{\|r^\star\|^2}{\sigma_{\min}^2(\mathbf{A})}.$$

For consistent systems the least-squares residual is $r^\star = 0$ and this bound guarantees exponential convergence in expectation at a rate $1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}$ [SV09]. For inconsistent systems, this bound only guarantees exponential convergence in expectation to within a convergence horizon $\|r^\star\|^2/\sigma_{\min}^2(\mathbf{A})$.

We derive a convergence result for Algorithm 6 which is similar to Equation (3.4) and leads to a better convergence rate and a smaller convergence horizon for inconsistent systems

when using uniform weights. To analyze the convergence, we begin by finding the update to the error at each iteration. Subtracting the exact solution $x^\star$ from both sides of the update rule in Equation (3.2) and using the fact that $\mathbf{A}_i e^k - r_i^\star = \mathbf{A}_i x^k - b_i$, we arrive at the error update

$$e^{k+1} = e^k - \frac{1}{q}\sum_{i\in\tau_k} w_i \frac{\mathbf{A}_i e^k - r_i^\star}{\|\mathbf{A}_i\|^2}\mathbf{A}_i^\top. \tag{3.5}$$

To simplify notation, we define the following matrices.

Definition 3.2.1 (Weighted Sampling Matrix)**.** *Define the* weighted sampling matrix

$$\mathbf{M}_k \overset{def}{=} \frac{1}{q}\sum_{i\in\tau_k} w_i \frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2},$$

where τ_k is a set of indices sampled independently from $\mathcal{D}$ with replacement and $\mathbf{I}$ is the identity matrix.

Using Definition 3.2.1, the error update from Equation (3.5) can be rewritten as

$$e^{k+1} = (\mathbf{I} - \mathbf{A}^\top \mathbf{M}_k \mathbf{A})e^k + \mathbf{A}^\top \mathbf{M}_k r^\star. \tag{3.6}$$

Definition 3.2.2 (Normalization, Probability, and Weight Matrices)**.** *Let* $\mathrm{Diag}\,(d_1, d_2, \ldots, d_m)$ *denote the diagonal matrix with $d_1, d_2, \ldots d_m$ on the diagonal. Define the* normalization matrix

$$\mathbf{D} \overset{def}{=} \mathrm{Diag}\,(\|\mathbf{A}_1\|, \|\mathbf{A}_2\|, \ldots, \|\mathbf{A}_m\|)$$

so that the matrix $\mathbf{D}^{-1}\mathbf{A}$ has rows with unit norm, the probability matrix

$$\mathbf{P} \overset{def}{=} \mathrm{Diag}\,(p_1, p_2, \ldots, p_m)\,,$$

where $p_j = \mathbb{P}(i = j)$ with $i \sim \mathcal{D}$, and the weight matrix

$$\mathbf{W} \overset{def}{=} \mathrm{Diag}\,(w_1, w_2, \ldots, w_m)\,.$$

The convergence analysis additionally relies on the expectations given in Lemma 3.2.1, whose proof can be found in Section 3.A.

Lemma 3.2.1. *Let* $\mathbf{M}_k, \mathbf{P}, \mathbf{W}$*, and* $\mathbf{D}$ *be defined as in Definitions 3.2.1 and 3.2.2. Then*

$$\mathbb{E}\left[\mathbf{M}_k\right] = \mathbf{PWD}^{-2}$$

and

$$\mathbb{E}\left[\mathbf{M}_k^\top \mathbf{A}\mathbf{A}^\top \mathbf{M}_k\right] = \frac{1}{q}\mathbf{PW}^2\mathbf{D}^{-2} + \left(1 - \frac{1}{q}\right)\mathbf{PWD}^{-2}\mathbf{AA}^\top\mathbf{PWD}^{-2}.$$

3.2.1 Coupling of Weights and Probabilities

Note that the weighted sampling matrix $\mathbf{M}_k$ is a sample mean, with the number of samples being the number of threads q. Thus, as the number of threads q goes to infinity, we have

$$\mathbf{M}_k \xrightarrow{q\to\infty} \mathbf{E}_{i\sim\mathcal{D}}\left[w_i \frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2}\right] = \mathbf{PWD}^{-2}.$$

Therefore, as we take more and more threads, the averaged RK update of Equation (3.3) approaches the deterministic update

$$x^{k+1} = (\mathbf{I} - \mathbf{A}^\top\mathbf{PWD}^{-2}\mathbf{A})x^k + \mathbf{A}^\top\mathbf{PWD}^{-2}b$$

and likewise the corresponding error update in Equation (3.6) approaches the deterministic update

$$e^{k+1} = (\mathbf{I} - \mathbf{A}^\top\mathbf{PWD}^{-2}\mathbf{A})e^k + \mathbf{A}^\top\mathbf{PWD}^{-2}r^\star.$$

Since we want the error of the limiting averaged RK method to converge to zero, we should require that this limiting error update have the zero vector as a fixed point. Thus, we require that

$$0 = \mathbf{A}^\top\mathbf{PWD}^{-2}r^\star$$

for any least-squares residual $r^\star$. This is guaranteed if Assumption 2 holds.

Assumption 2. *The probability matrix* $\mathbf{P}$ *and weight matrix* $\mathbf{W}$ *are chosen to satisfy*

$$\mathbf{PWD}^{-2} = \alpha\mathbf{I}$$

for some scalar relaxation parameter $\alpha > 0$*.*

3.2.2 General Weights

We now state a general convergence result for RK with averaging in Theorem 3.2.2. The proof is given in Section 3.B. Theorem 3.2.2 in its general form is difficult to interpret, so we defer a detailed analysis to Section 3.3 in which the assumption of uniform weights simplifies the bound significantly.

Theorem 3.2.2. *Suppose* $\mathbf{P}$ *and* $\mathbf{W}$ *of Definition 3.2.2 are chosen such that* $\mathbf{PWD}^{-2} = \frac{\alpha}{\|\mathbf{A}\|_F^2}\mathbf{I}$ *for relaxation parameter* $\alpha > 0$*. Then the error at each iteration of Algorithm 6 satisfies*

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] \leq \sigma_{\max}\left(\left(\mathbf{I} - \alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 - \frac{\alpha^2}{q}\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2\right)\|e^k\|^2 + \frac{\alpha}{q}\frac{\|r^k\|_{\mathbf{W}}^2}{\|\mathbf{A}\|_F^2},$$

where $r^k \overset{def}{=} b - \mathbf{A}x^k$ *is the residual of the* k^{th} *iterate,* $\|\cdot\|_{\mathbf{W}}^2 = \langle\cdot, \mathbf{W}\cdot\rangle$ *and* $\|\mathbf{A}\|_F^2 = \sum_{i,j}\mathbf{A}_{ij}^2$*.*

Here, and for the remainder of the chapter, we take the expectation $\mathbb{E}\left[\|e^{k+1}\|^2\right]$ conditioned on e_k.

As we shall see in Section 3.3, the relaxation parameter α and number of threads q are closely tied to both the convergence horizon and convergence rate. The convergence horizon is proportional to $\frac{\alpha^2}{q}$, so smaller α and larger q lead to a smaller convergence horizon. Increasing the value of α improves the convergence rate of the algorithm up to a critical point beyond which further increasing α leads to slower convergence rates. Increasing the number of threads q improves the convergence rate, $\frac{1}{q}$-asymptotically approaching an optimal rate as $q \to \infty$.

3.3 Uniform Weights

We can simplify the analysis significantly if we assume that $\mathbf{W} = \alpha\mathbf{I}$, or equivalently that the weights are uniform. In this case, the update for each iteration becomes

$$x^{k+1} = x^k - \frac{\alpha}{q}\sum_{i\in\tau_k}\frac{\mathbf{A}_i x^k - b_i}{\|\mathbf{A}_i\|^2}\mathbf{A}_i^\top,$$

where $i \in \tau_k$ are independent samples from $\mathcal{D}$ with $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$. Under these conditions, the expected error bound of Theorem 3.2.2 can be simplified to remove the dependence on

r^k. This simplification leads to the more interpretable error bound given in Corollary 1. In particular, increasing q leads to both a faster convergence rate and smaller convergence horizon. If the relaxation parameter is chosen as $\alpha = 1$ and a single row is selected at each iteration, we arrive at the RK method [SV09]. Using a relaxation parameter $\alpha \neq 1$ results in the relaxed RK method [HN90b, HN90a].

Corollary 1. *Suppose* $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$ *and* $\mathbf{W} = \alpha\mathbf{I}$. *Then the expected error at each iteration of Algorithm 6 satisfies*

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] \leq \sigma_{\max}\left(\left(\mathbf{I} - \alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 + \frac{\alpha^2}{q}\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\alpha^2\|r^\star\|^2}{q\|\mathbf{A}\|_F^2}.$$

The proof of Corollary 1 follows immediately from Theorem 3.2.2 and can be found in Subsection 3.D.1.

3.3.0.1 Randomized Kaczmarz

If a single row is chosen at each iteration, with $\mathbf{W} = \mathbf{I}$ and $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$, then Algorithm 6 becomes the version of RK stated in [SV09]. In this case,

$$\|r^k\|_{\mathbf{W}}^2 = \|\mathbf{A}e^k\|^2 + \|r^\star\|^2. \tag{3.7}$$

Applying Theorem 3.2.2 leads to the following corollary, which recovers the error bound in Equation (3.4).

Corollary 2. *Suppose* $q = 1$, $\mathbf{W} = \mathbf{I}$ *and* $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$. *Then the expected error at each iteration of Algorithm 6 satisfies*

$$\begin{aligned}\mathbb{E}\left[\|e^{k+1}\|^2\right] &\leq \sigma_{\max}\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2} \\ &= \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2}.\end{aligned}$$

A proof of Corollary 2 is included in Subsection 3.D.2.

3.4 Consistent Systems

For consistent systems, Algorithm 6 converges to the solution $x^\star$ exponentially in expectation with the following guaranteed convergence rate.

Corollary 3. *Suppose* $\mathbf{P}$ *and* $\mathbf{W}$ *of Definition 3.2.2 are chosen such that* $\mathbf{PWD}^{-2} = \frac{\alpha}{\|\mathbf{A}\|_F^2}\mathbf{I}$ *for some constant* $\alpha > 0$*. Then the error at each iteration of Algorithm 6 satisfies*

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] \leq \sigma_{\max}\left(\left(\mathbf{I} - \alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 + \frac{\mathbf{A}^\top}{\|\mathbf{A}\|_F}\left(\frac{\alpha}{q}\mathbf{W} - \frac{\alpha^2}{q}\frac{\mathbf{A}\mathbf{A}^\top}{\|\mathbf{A}\|_F^2}\right)\frac{\mathbf{A}}{\|\mathbf{A}\|_F}\right)\|e^k\|^2.$$

Corollary 3 results from taking $r^\star = 0$ in the proof of Theorem 3.2.2.

3.5 Suggested Relaxation Parameter α for Consistent Systems With Uniform Weights

For consistent systems and using uniform weights, Algorithm 6 becomes a subcase of the parallel sketch-and-project method described by Richtárik and Takáč [RT20]. They suggest a choice of relaxation parameter

$$\alpha = \frac{q}{1 + (q-1)\frac{\sigma_{\max}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}}, \tag{3.8}$$

which is chosen to optimize their convergence rate guarantee.

Analogously, for uniform weights, we can calculate the value of α to minimize the bound given in Corollary 1.

Theorem 3.5.1. *Suppose* $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$ *and* $\mathbf{W} = \alpha\mathbf{I}$*. Then, the relaxation parameter* α *which yields the fastest convergence rate guarantee in Corollary 1 is*

$$\alpha^\star = \begin{cases} \frac{q}{1+(q-1)s_{\min}}, & 1-(q-1)(s_{\max}-s_{\min}) \geq 0, \\ \frac{2q}{1+(q-1)(s_{\min}+s_{\max})}, & 1-(q-1)(s_{\max}-s_{\min}) < 0 \end{cases}$$

where $s_{\min} = \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}$ *and* $s_{\max} = \frac{\sigma_{\max}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}$.

The proof of this result can be found in Section 3.C.

When $q = 1$, the second condition cannot hold, so only the first formula is used. Plugging in $q = 1$, the term that depends on $s_{\min}$ vanishes and we get that $\alpha^\star = 1$. When $q > 1$, we can divide by $q - 1$ and express the condition in terms of the spectral gap as $s_{\max} - s_{\min} \leq \frac{1}{q-1}$. For matrices where the spectral gap is positive, we can also view this as a condition on the number of threads, $q \leq 1 + \frac{1}{s_{\max} - s_{\min}}$. We see that for low numbers of threads, we expect

$$\alpha^\star = \frac{q}{1 + (q-1)s_{\min}}$$

while for high numbers of threads, we expect

$$\alpha^\star = \frac{2q}{1 + (q-1)\left(s_{\min} + s_{\max}\right)}.$$

Note that this differs from the relaxation parameter α suggested by Richtárik and Takáč [RT20], given in Equation (3.8). This is due to the fact that our convergence rate guarantee is tighter, and thus we expect that our suggested relaxation parameter α should be closer to the truly optimal value. We compare these two choices of the relaxation parameter α experimentally in Subsection 3.6.3 and show that our suggested relaxation parameter $\alpha^\star$ is indeed closer to the true optimal value, especially for large numbers of threads q.

3.6 Experiments

We present several experiments to demonstrate the convergence of Algorithm 6 under various conditions. In particular, we study the effects of the number of threads q, the relaxation parameter α, the weight matrix $\mathbf{W}$, and the probability matrix $\mathbf{P}$.

3.6.1 Procedure

For each experiment, we run 100 independent trials each starting with the initial iterate $x^0 = 0$ and average the squared error norms $\|e^k\|^2$ across the trials. We sample $\mathbf{A}$ from 100×10 standard Gaussian matrices and least-squares solution $x^\star$ from 10-dimensional

standard Gaussian vectors, normalized so that $\|x^\star\| = 1$. To form inconsistent systems, we generate the least-squares residual $r^\star$ as a Gaussian vector orthogonal to the range of $\mathbf{A}$, also normalized so that $\|r^\star\| = 1$. Finally, b is computed as $r^\star + \mathbf{A}x^\star$. While small, these examples are useful to illustrate the behavior of Algorithm 6 as it relates to its parameters.

3.6.2 The Effect of the Number of Threads

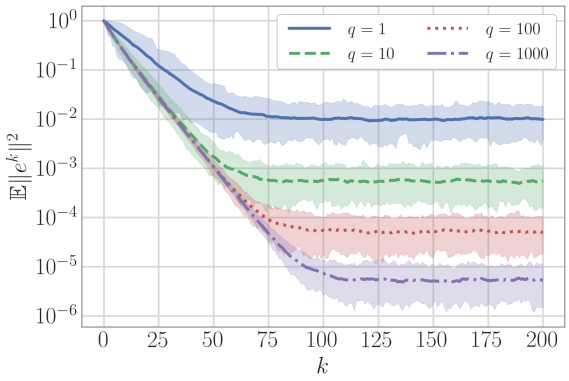
In Figure 3.1, we see the effects of the number of threads q on the approximation error of Algorithm 6 for different choices of the weight matrices $\mathbf{W}$ and probability matrices $\mathbf{P}$. In Figures 3.1a and 3.1b, $\mathbf{W}$ and $\mathbf{P}$ satisfy Assumption 2, while in Figure 3.1c they do not.

In Figures 3.1a and 3.1b, as the number of threads q increases by a factor of ten, we see a corresponding decrease in the magnitude of the convergence horizon by approximately the same factor. This result corroborates what we expect based on Theorem 3.2.2 and Corollary 1. For Figure 3.1c, we do not see the same consistent decrease in the magnitude of the convergence horizon. As q increases, for weight matrices $\mathbf{W}$ and probability matrices $\mathbf{P}$ that do not satisfy Assumption 2, the iterates x^k approach a weighted least-squares solution instead of the desired least-squares solution $x^\star$ (see Subsection 3.2.1).

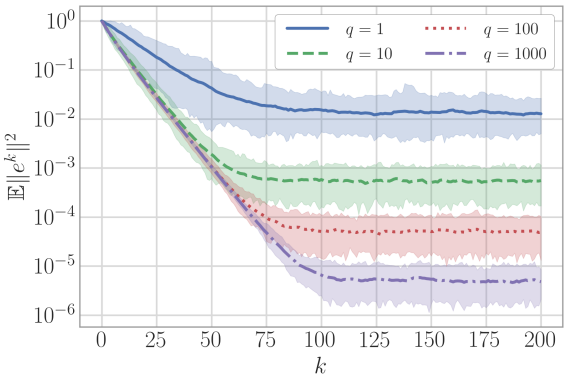
The rate of convergence in Figure 3.1 also improves as the number of threads q increases. As q increases, we see diminishing returns in the convergence rate. We expect this behavior based on the dependence on $\frac{1}{q}$ in Theorem 3.2.2 and Corollary 1.

3.6.3 The Effect of the Relaxation Parameter α

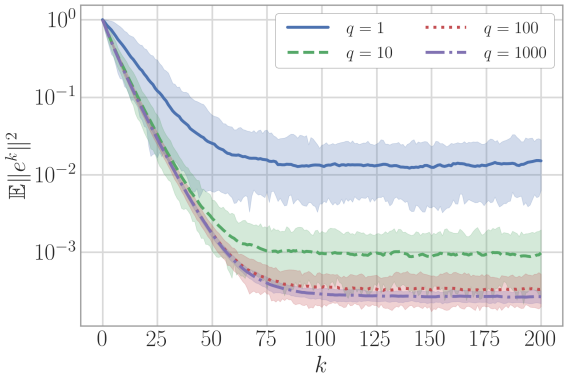
In Figure 3.2, we observe the effect on the convergence rate and convergence horizon as we vary the relaxation parameter α. From Theorem 3.2.2, we expect that the convergence horizon increases with α and indeed observe this experimentally. The squared norms of the errors behave similarly as α varies for both sets of weights and probabilities considered, each of which satisfy Assumption 2.



(a) Uniform weights $w_i = 1$ and probabilities proportional to squared row norms $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$.

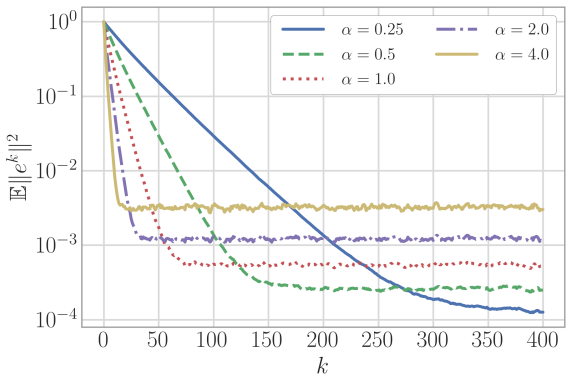


(b) Weights proportional to squared row norms $w_i = m\frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$ and uniform probabilities $p_i = \frac{1}{m}$.

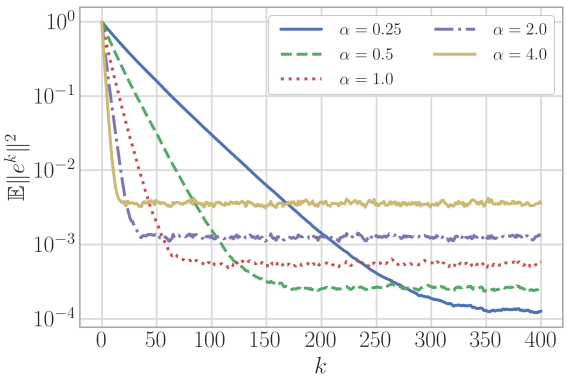


(c) Uniform weights $w_i = 1$ and uniform probabilities $p_i = \frac{1}{m}$.

Figure 3.1: The effect of the number of threads on the average squared error norm vs iteration for Algorithm 6 applied to inconsistent systems. The weights w_i and probabilities p_i in a and b satisfy Assumption 2, while in c they do not. Shaded areas indicate the middle 90% performance, measured over 100 trials.



(a) Uniform weights $w_i = \alpha$, probabilities proportional to squared row norms $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$, and number of threads $q = 10$.

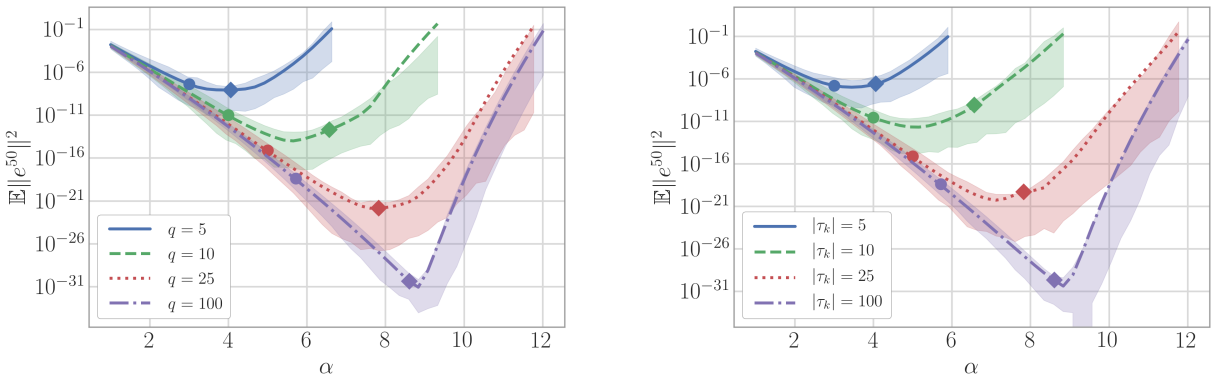


(b) Weights proportional to squared row norms $w_i = \alpha m \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$, uniform probabilities $p_i = \frac{1}{m}$, and number of threads $q = 10$.

Figure 3.2: The effect of the relaxation parameter α on the average squared error norm vs iteration for Algorithm 6 applied to inconsistent systems.

For larger values of the relaxation parameter α, the convergence rate for Algorithm 6 eventually decreases and the method can ultimately diverge. This behavior can be seen in Figure 3.3, which plots the squared error norm after 100 iterations for consistent Gaussian systems, various α, and various numbers of threads q. In Figure 3.3a, we use uniform weights $w_i = \alpha$ with probabilities proportional to the squared row norms $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$, and in Figure 3.3b, we use weights proportional to the row norms $w_i = \alpha m \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$ with uniform probabilities $p_i = \frac{1}{m}$.

For each value of q, we plot two markers on the curve at the estimated optimal values of α. The diamond markers are estimates computed using Theorem 3.5.1, and the circle markers are estimates using the formula from Richtárik and Takáč [RT20]. These values are also contained in Table 3.1. In terms of the number of iterations required, we find that the true optimal value increases with q. Comparing the α values from [RT20] with the α that minimize the curves in Figure 3.3, we find that they generally underestimate the true optimal α. In comparison, the estimates calculated using Theorem 3.5.1 are much closer to the observed optimal values of α, especially for high q



(a) $w_i = \alpha$, $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$.

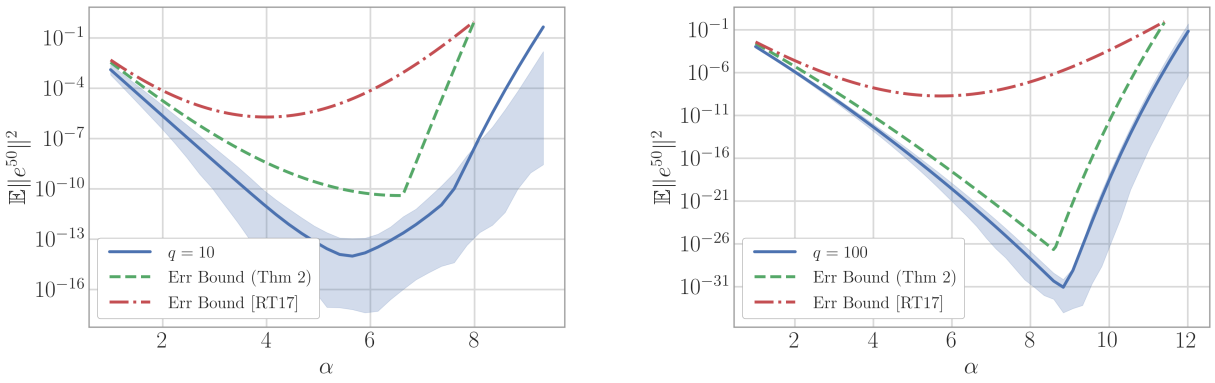
(b) $w_i = \alpha m \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$, $p_i = \frac{1}{m}$.

Figure 3.3: Squared error norm after 50 iterations of Algorithm 6 on consistent systems for various choices of relaxation parameter α. Shaded areas indicate the middle 90% performance, measured over 100 trials. Diamond markers are estimates of the optimal alpha using Theorem 3.5.1, and circle markers are estimates using the formula from Richtárik and Takáč [RT20]

Table 3.1: Calculated optimal $\alpha^\star$ for matrix $\mathbf{A}$ used in Figure 3.3a.

	$q = 5$	$q = 10$	$q = 25$	$q = 100$
α (Eqn 3.8) [Richtárik et al.]	3.00	4.00	5.00	5.72
$\alpha^\star$ (our Theorem 3.5.1)	4.06	6.57	7.83	8.61

We believe that the main reason our estimate of $\alpha^\star$ is closer than that of Richtárik and Takáč [RT20] is due to our bound being relatively tighter than Equation (3.8). In Figures 3.4a and 3.4b, we plot the error bounds produced by Equation (3.8) and Theorem 3.5.1 after 50 iterations for $q = 10$ and $q = 100$. We observe that as the number of threads increases, our bound approaches the empirical result, while the bound of Richtárik and Takáč [RT20] does not.



(a) $q = 10$.

(b) $q = 100$.

Figure 3.4: Squared error norm after 50 iterations of Algorithm 6 on consistent systems for various choices of relaxation parameter α. Uniform weights $w_i = \alpha$ and probabilities proportional to squared row norms $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$.

3.7 Conclusion

We proved a general error bound for RK with averaging given in Algorithm 6 in terms of the number of threads q and a relaxation parameter α. We found a coupling between the probability matrix $\mathbf{P}$ and the weight matrix $\mathbf{W}$ that leads to a reduced convergence horizon. We demonstrated that for uniform weights, i.e. $\mathbf{W} \propto \mathbf{I}$, the rate of convergence and convergence horizon for Algorithm 6 improve both in theory and practice as the number of threads q increases. Based on the error bound, we also derived an optimal value for the relaxation parameter α which increases convergence speed, and compared with existing results.

3.A Proof of Lemma 3.2.1

Proof. Expanding the definition of the weighted sampling matrix $\mathbf{M}_k$ as a weighted average of the i.i.d. sampling matrices $\frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2}$, we see that

$$\mathbb{E}\left[\mathbf{M}_k\right] = \mathbb{E}\left[\frac{1}{q}\sum_{i\in\tau_k} w_i \frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2}\right] = \mathbb{E}\left[w_i \frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2}\right] = \sum_{i=1}^{m} p_i w_i \frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2} = \mathbf{P}\mathbf{W}\mathbf{D}^{-2}.$$

Likewise, we can compute

$$\begin{aligned}
\mathbb{E}\left[\mathbf{M}_k^\top \mathbf{A}\mathbf{A}^\top \mathbf{M}_k\right] &= \mathbb{E}\left[\left(\frac{1}{q}\sum_{i\in\tau_k} w_i \frac{\mathbf{I}_i^\top \mathbf{A}_i}{\|\mathbf{A}_i\|^2}\right)\left(\frac{1}{q}\sum_{j\in\tau_k} w_j \frac{\mathbf{A}_j^\top \mathbf{I}_j}{\|\mathbf{A}_j\|^2}\right)\right] \\
&= \frac{1}{q}\mathbb{E}\left[\left(w_i \frac{\mathbf{I}_i^\top \mathbf{A}_i}{\|\mathbf{A}_i\|^2}\right)\left(w_i \frac{\mathbf{A}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2}\right)\right] + (1-\frac{1}{q})\mathbb{E}\left[w_i \frac{\mathbf{I}_i^\top \mathbf{A}_i}{\|\mathbf{A}_i\|^2}\right]\mathbb{E}\left[w_j \frac{\mathbf{A}_j^\top \mathbf{I}_j}{\|\mathbf{A}_j\|^2}\right] \\
&= \frac{1}{q}\mathbb{E}\left[w_i^2 \frac{\mathbf{I}_i^\top \mathbf{I}_i}{\|\mathbf{A}_i\|^2}\right] + \left(1-\frac{1}{q}\right)\mathbf{P}\mathbf{W}\mathbf{D}^{-2}\mathbf{A}\mathbf{A}^\top \mathbf{P}\mathbf{W}\mathbf{D}^{-2} \\
&= \frac{1}{q}\mathbf{P}\mathbf{W}^2\mathbf{D}^{-2} + \left(1-\frac{1}{q}\right)\mathbf{P}\mathbf{W}\mathbf{D}^{-2}\mathbf{A}\mathbf{A}^\top \mathbf{P}\mathbf{W}\mathbf{D}^{-2}
\end{aligned}$$

by separating the cases where $i = j$ from those where $i \neq j$ and utilizing the independence of the indices sampled in τ_k. □

3.B Proof of Theorem 3.2.2

Proof. We prove Theorem 3.2.2 starting from from the error update in Equation (3.6). Expanding the squared error norm,

$$\begin{aligned}
\|e^{k+1}\|^2 &= \|(\mathbf{I} - \mathbf{A}^\top \mathbf{M}_k \mathbf{A})e^k + \mathbf{A}^\top \mathbf{M}_k r^\star\|^2 \\
&= \|(\mathbf{I} - \mathbf{A}^\top \mathbf{M}_k \mathbf{A})e^k\|^2 + 2\langle(\mathbf{I} - \mathbf{A}^\top \mathbf{M}_k \mathbf{A})e^k, \mathbf{A}^\top \mathbf{M}_k r^\star\rangle + \|\mathbf{A}^\top \mathbf{M}_k r^\star\|^2.
\end{aligned}$$

Upon taking expections, the middle term simplifies since $\mathbf{A}^\top \mathbb{E}\left[\mathbf{M}_k\right] r^\star = 0$ by Assumption 2. Thus,

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] = \mathbb{E}\left[\|(\mathbf{I} - \mathbf{A}^\top \mathbf{M}_k \mathbf{A})e^k\|^2\right] - 2\mathbb{E}\left[\langle \mathbf{A}^\top \mathbf{M}_k \mathbf{A} e^k, \mathbf{A}^\top \mathbf{M}_k r^\star\rangle\right] + \mathbb{E}\left[\|\mathbf{A}^\top \mathbf{M}_k r^\star\|^2\right]. \tag{3.9}$$

Making use of Lemma 3.2.1 to take the expectation of the first term in Equation (3.9),

$$\begin{aligned}
&\mathbb{E}\left[\|(\mathbf{I}-\mathbf{A}^\top\mathbf{M}_k\mathbf{A})e^k\|^2\right] \\
&\quad= \mathbb{E}\left[\left\langle e^k, (\mathbf{I}-\mathbf{A}^\top\mathbf{M}_k\mathbf{A})^\top(\mathbf{I}-\mathbf{A}^\top\mathbf{M}_k\mathbf{A})e^k\right\rangle\right] \\
&\quad= \left\langle e^k, (\mathbf{I}-2\mathbf{A}^\top\mathbb{E}\left[\mathbf{M}_k\right]\mathbf{A}+\mathbf{A}^\top\mathbb{E}\left[\mathbf{M}_k^\top\mathbf{A}\mathbf{A}^\top\mathbf{M}_k\right]\mathbf{A})e^k\right\rangle \\
&\quad= \left\langle e^k, \left(\mathbf{I}-2\alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}+\frac{\alpha}{q}\frac{\mathbf{A}^\top\mathbf{W}\mathbf{A}}{\|\mathbf{A}\|_F^2}+\alpha^2\left(1-\frac{1}{q}\right)\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2\right)e^k\right\rangle \\
&\quad= \left\langle e^k, \left(\left(\mathbf{I}-\alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2+\frac{\mathbf{A}^\top}{\|\mathbf{A}\|_F}\left(\frac{\alpha}{q}\mathbf{W}-\frac{\alpha^2}{q}\frac{\mathbf{A}\mathbf{A}^\top}{\|\mathbf{A}\|_F^2}\right)\frac{\mathbf{A}}{\|\mathbf{A}\|_F}\right)e^k\right\rangle.
\end{aligned}$$

Since $\mathbf{A}^\top r^\star = 0$, for the second term,

$$\begin{aligned}
2\mathbb{E}\left[\langle\mathbf{A}^\top\mathbf{M}_k\mathbf{A}e^k, \mathbf{A}^\top\mathbf{M}_k r^\star\rangle\right] &= 2\langle\mathbf{A}e^k, \mathbb{E}\left[\mathbf{M}_k^\top\mathbf{A}\mathbf{A}^\top\mathbf{M}_k\right]r^\star\rangle \\
&= 2\frac{\alpha}{q\|\mathbf{A}\|_F^2}\langle\mathbf{A}e^k, \mathbf{W}r^\star\rangle.
\end{aligned}$$

Similarly, for the last term,

$$\mathbb{E}\left[\|\mathbf{A}^\top\mathbf{M}_k r^\star\|^2\right] = \frac{\alpha}{q}\frac{\|r^\star\|_{\mathbf{W}}^2}{\|\mathbf{A}\|_F^2}.$$

Combining these in Equation (3.9),

$$\begin{aligned}
\mathbb{E}\left[\|e^{k+1}\|^2\right] &= \left\langle e^k, \left(\mathbf{I}-\alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 e^k\right\rangle \\
&\quad+\left\langle e^k, \frac{\mathbf{A}^\top}{\|\mathbf{A}\|_F^2}\left(\frac{\alpha}{q}\mathbf{W}-\frac{\alpha^2}{q}\frac{\mathbf{A}\mathbf{A}^\top}{\|\mathbf{A}\|_F^2}\right)\mathbf{A}e^k\right\rangle - 2\frac{\alpha}{q}\frac{\langle\mathbf{A}e^k, \mathbf{W}r^\star\rangle}{\|\mathbf{A}\|_F^2}+\frac{\alpha}{q}\frac{\|r^\star\|_{\mathbf{W}}^2}{\|\mathbf{A}\|_F^2} \\
&= \left\langle e^k, \left(\left(\mathbf{I}-\alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2-\frac{\alpha^2}{q}\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2\right)e^k\right\rangle+\frac{\alpha}{q}\frac{\|r^k\|_{\mathbf{W}}^2}{\|\mathbf{A}\|_F^2} \\
&\leq \sigma_{\max}\left(\left(\mathbf{I}-\alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2-\frac{\alpha^2}{q}\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2\right)\|e^k\|^2+\frac{\alpha}{q}\frac{\|r^k\|_{\mathbf{W}}^2}{\|\mathbf{A}\|_F^2}.
\end{aligned}$$

□

3.C Proof of Theorem 3.5.1

Proof. We seek to optimize the convergence rate constant from Corollary 1,

$$\sigma_{\max}\left(\left(\mathbf{I}-\alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2+\frac{\alpha^2}{q}\left(\mathbf{I}-\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)$$

with respect to α. To do this, we first simplify from a matrix polynomial to a maximum over scalar polynomials in α with coefficients based on each singular value of $\mathbf{A}$. We then show that the maximum occurs when either the minimum or maximum singular value of $\mathbf{A}$ is used. Finally, we derive a condition for which singular value to use, and determine a value of α that minimizes the maximum singular value.

Defining $\mathbf{Q}^\top\Sigma\mathbf{Q}=\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}$ as the eigendecomposition, and the polynomial

$$p(\sigma)\overset{\text{def}}{=}1-2\alpha\sigma+\alpha^2\left(\frac{\sigma}{q}+\left(1-\frac{1}{q}\right)\sigma^2\right),$$

the convergence rate constant from Corollary 1 can be written as $\sigma_{\max}\left(p\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\right)$. Since $p\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)$ is a polynomial of a symmetric matrix, its singular vectors are the same as those of its argument, while its corresponding singular values are the polynomial p applied to the singular values of the original matrix. That is,

$$p\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)=p\left(\mathbf{Q}^\top\Sigma\mathbf{Q}\right)=\mathbf{Q}^\top p\left(\Sigma\right)\mathbf{Q}.$$

Thus, the convergence rate constant can be written as

$$\sigma_{\max}\left(p\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\right)=\sigma_{\max}\left(p(\Sigma)\right).$$

Moreover, we can bound this extremal singular value by the maximum of the polynomial p over an interval containing the spectrum of Σ

$$\sigma_{\max}\left(p\left(\Sigma\right)\right)\leq\max\left|p\left(\sigma\right)\right|\quad\text{subject to}\quad\sigma\in\left[s_{\min},s_{\max}\right].$$

Here, the singular values of Σ are bounded from below by $s_{\min}\overset{\text{def}}{=}\frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}$ and above by $s_{\max}\overset{\text{def}}{=}\frac{\sigma_{\max}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}$ since Σ is the diagonal matrix of singular values of $\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}$. Note that the

polynomial can be factored as $p(\sigma) = (1-\sigma\alpha)^2 + \frac{\sigma\alpha^2}{q}(1-\sigma)$, and is positive for $\sigma \in [0,1]$, which contains $[s_{\min}, s_{\max}]$. Also, since the coefficient of the σ^2 term of the polynomial p is $\alpha^2\left(1-\frac{1}{q}\right)$, and $\alpha^2\left(1-\frac{1}{q}\right) \geq 0$, the polynomial is convex in σ on the interval $[s_{\min}, s_{\max}]$. Thus, the maximum of p on the interval $[s_{\min}, s_{\max}]$ is attained at one of the two endpoints $s_{\min}, s_{\max}$ and we have the bound

$$\sigma_{\max}\left(p\left(\Sigma\right)\right) = \max\left(p\left(s_{\min}\right), p\left(s_{\max}\right)\right).$$

To optimize this bound with respect to α, we first find conditions on α such that $p(s_{\min}) < p(s_{\max})$. If $s_{\max} = s_{\min}$, this obviously never holds; otherwise, $s_{\max} > s_{\min}$ and

$$1 - 2\alpha s_{\min} + \alpha^2\left[\frac{s_{\min}}{q} + \left(1-\frac{1}{q}\right)s_{\min}^2\right] < 1 - 2\alpha s_{\max} + \alpha^2\left[\frac{s_{\max}}{q} + \left(1-\frac{1}{q}\right)s_{\max}^2\right].$$

Grouping like terms and cancelling, we get

$$\alpha\left(2-\frac{\alpha}{q}\right)\left(s_{\max}-s_{\min}\right) < \alpha^2\left(1-\frac{1}{q}\right)\left(s_{\max}^2 - s_{\min}^2\right).$$

Since $\frac{\alpha}{q} > 0$, we can divide it from both sides to find

$$\left(2q-\alpha\right)\left(s_{\max}-s_{\min}\right) < \alpha\left(q-1\right)\left(s_{\max}^2 - s_{\min}^2\right).$$

Since $s_{\max} > s_{\min}$, we can divide both sides by $s_{\max} - s_{\min}$ and simplify to

$$\begin{aligned} 2q - \alpha &< \alpha\left(q-1\right)\left(s_{\max}+s_{\min}\right) \\ 2q &< \alpha\left(1+\left(q-1\right)\left(s_{\max}+s_{\min}\right)\right) \\ \alpha &> \frac{2q}{1+\left(q-1\right)\left(s_{\min}+s_{\max}\right)} \stackrel{\text{def}}{=} \widehat{\alpha}. \end{aligned}$$

Thus, we see

$$\sigma_{\max}\left(p\left(\Sigma\right)\right) = \begin{cases} p\left(s_{\max}\right), & \alpha \geq \widehat{\alpha} \\ p\left(s_{\min}\right), & \alpha < \widehat{\alpha} \end{cases}$$

For the first term,

$$\begin{aligned} \frac{\partial}{\partial\alpha}p(s_{\max}) &= -2s_{\max} + 2\left(\frac{s_{\max}}{q} + \left(1-\frac{1}{q}\right)s_{\max}^2\right)\alpha \\ &\geq -2s_{\max} + 2\left(\frac{s_{\max}}{q} + \left(1-\frac{1}{q}\right)s_{\max}^2\right)\widehat{\alpha} \end{aligned}$$

since $\alpha \leq \widehat{\alpha}$ and the coefficient is positive. Factoring $\frac{2s_{\max}}{q}$ from the second term and substituting for $\widehat{\alpha}$, we get

$$
\begin{aligned}
&= -2s_{\max} + \frac{2s_{\max}}{q}\left(1 + (q-1)\, s_{\max}\right)\widehat{\alpha} \\
&= -2s_{\max} + \frac{2s_{\max}}{q}\left(1 + (q-1)\, s_{\max}\right)\frac{2q}{1 + (1-q)\left(s_{\min} + s_{\max}\right)} \\
&= -2s_{\max} + 2s_{\max}\frac{2\left(1 + (q-1)s_{\max}\right)}{1 + (q-1)(s_{\max} + s_{\min})} \\
&= 2s_{\max}\left[-1 + \frac{2\left(1 + (q-1)s_{\max}\right)}{1 + (q-1)(s_{\max} + s_{\min})}\right] \\
&= 2s_{\max}\left[\frac{1 + (q-1)(s_{\max} - s_{\min})}{1 + (q-1)(s_{\max} + s_{\min})}\right] \\
&> 0
\end{aligned}
$$

since all terms in both numerator and denominator are positive. Thus, the function is monotonic increasing on $\alpha \in [\widehat{\alpha}, \infty)$, and the minimum is at the lower endpoint $\alpha^\star = \widehat{\alpha}$. Similarly, for the second term,

$$
\begin{aligned}
\frac{\partial}{\partial\alpha}p(s_{\min}) &= -2s_{\min} + 2\left(\frac{s_{\min}}{q} + \left(1 - \frac{1}{q}\right)s_{\min}^2\right)\alpha \\
&< -2s_{\min} + 2\left(\frac{s_{\min}}{q} + \left(1 - \frac{1}{q}\right)s_{\min}^2\right)\widehat{\alpha} \\
&= 2s_{\min}\left[\frac{1 - (q-1)(s_{\max} - s_{\min})}{1 + (q-1)(s_{\max} + s_{\min})}\right].
\end{aligned}
$$

If

$$
1 - (q-1)(s_{\max} - s_{\min}) < 0, \tag{3.10}
$$

this function is monotonic decreasing on $\alpha \in (-\infty, \alpha^\star]$, and the minimum is at the upper endpoint $\alpha = \alpha^\star$. Otherwise, the minimum occurs at the critical point, so we set the derivative

to 0 and solve for $\alpha^\star$

$$\begin{aligned}
\frac{\partial}{\partial\alpha}p(s_{\min}) &= -2s_{\min} + 2\left(\frac{s_{\min}}{q} + \left(1-\frac{1}{q}\right)s_{\min}^2\right)\alpha^\star \\
&= -2s_{\min} + \frac{2s_{\min}}{q}\left(1+(q-1)\,s_{\min}\right)\alpha^\star \\
&= 0 \\
\frac{2s_{\min}}{q}\left(1+(q-1)\,s_{\min}\right)\alpha^\star &= 2s_{\min} \\
\alpha^\star &= \frac{q}{1+(q-1)s_{\min}}.
\end{aligned}$$

□

3.D Corollary Proofs

We provide proofs for the corollaries of Section 3.2, which follow from Theorem 3.2.2.

3.D.1 Proof of Corollary 1

Proof. Suppose $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$ and $\mathbf{W} = \alpha\mathbf{I}$. From the proof of Theorem 3.2.2,

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] = \left\langle e^k, \left(\left(\mathbf{I} - \alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 - \frac{\alpha^2}{q}\left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2\right)e^k\right\rangle + \frac{\alpha}{q}\frac{\|r^k\|_{\mathbf{W}}^2}{\|\mathbf{A}\|_F^2}.$$

In this case, since $\mathbf{A}^\top r^\star = 0$, $\langle \mathbf{A}e^k, r^\star\rangle = 0$ and

$$\begin{aligned}
\|r^k\|_{\mathbf{W}}^2 &= \alpha\|\mathbf{A}e^k\|^2 + 2\alpha\langle\mathbf{A}e^k, r^\star\rangle + \alpha\|r^\star\|^2 \\
&= \alpha\langle e^k, \mathbf{A}^\top\mathbf{A}e^k\rangle + \alpha\|r^\star\|^2.
\end{aligned}$$

Combining the two expressions above, we arrive at the desired result

$$\begin{aligned}
\mathbb{E}\left[\|e^{k+1}\|^2\right] &= \left\langle e^k, \left(\left(\mathbf{I} - \alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 + \frac{\alpha^2}{q}\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)e^k\right\rangle + \frac{\alpha^2\|r^\star\|^2}{q\|\mathbf{A}\|_F^2} \\
&\leq \sigma_{\max}\left(\left(\mathbf{I} - \alpha\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 + \frac{\alpha^2}{q}\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\alpha^2\|r^\star\|^2}{q\|\mathbf{A}\|_F^2}.
\end{aligned}$$

□

3.D.2 Proof of Corollary 2

Proof. Suppose $q = 1$, $\mathbf{W} = \mathbf{I}$ and $p_i = \frac{\|\mathbf{A}_i\|^2}{\|\mathbf{A}\|_F^2}$.

$$\begin{aligned}\mathbb{E}\left[\|e^{k+1}\|^2\right] &\leq \sigma_{\max}\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2} \\ &= \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2}.\end{aligned}$$

From the proof of Theorem 3.2.2,

$$\mathbb{E}\left[\|e^{k+1}\|^2\right] = \left\langle e^k, \left(\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 - \left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2\right)e^k\right\rangle + \frac{\|r^k\|^2}{\|\mathbf{A}\|_F^2}.$$

Decomposing r^k,

$$\begin{aligned}\|r^k\|^2 &= \|\mathbf{A}e^k\|^2 + \|r^\star\|^2 \\ &= \langle e^k, \mathbf{A}^\top\mathbf{A}e^k\rangle + \|r^\star\|^2.\end{aligned}$$

Combining the expressions above, we arrive at the desired result

$$\begin{aligned}\mathbb{E}\left[\|e^{k+1}\|^2\right] &= \left\langle e^k, \left(\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 - \left(\frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)^2 + \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)e^k\right\rangle + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2} \\ &= \left\langle e^k, \left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)e^k\right\rangle + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2} \\ &\leq \sigma_{\max}\left(\mathbf{I} - \frac{\mathbf{A}^\top\mathbf{A}}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2} \\ &= \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\|\mathbf{A}\|_F^2}\right)\|e^k\|^2 + \frac{\|r^\star\|^2}{\|\mathbf{A}\|_F^2}.\end{aligned}$$

□

CHAPTER 4

Subgraph Matching on Multiplex Networks*

4.1 Introduction

Multiplex networks [KAB14] are increasingly useful data structures for representing entities and their interactions in disciplines such as bioinformatics [ZL17], social networks [Ver79], ecological networks [PPP17], and neural networks [BBB16]. Here, we focus on the special case of multiplex networks consisting of labeled directed multigraphs (Definition 4.1.1). *Subgraph matching* is the process of determining whether a given pattern called a *template* network occurs as a subgraph of a larger *world* network, and if so, exactly where it occurs and how many times [CFS03].

Subgraph matching is commonly used in bioinformatics [ZLY09], social network analysis [Fan12, YG13], and other applications [CFS03]. It is also an important subroutine in frequent subgraph mining [YH02, HWP03] and graph database search [ZMC11]. Despite the abundance of multiplex network data in these applications, there are relatively few subgraph matching algorithms that expressly support multiplex networks [IIP16, MBF20] compared to the number of algorithms that support single-channel networks [Ull76, CFS04, CFS17, CFS18, Sol10, HLL13, BCL16]. In this chapter, we introduce a new algorithm for subgraph matching on multiplex networks and discuss some simplifications of the subgraph matching problem.

We refer to any subgraph of the world that matches the template as a *signal*. For sufficiently simple templates, there are efficient algorithms for counting and listing all corresponding

*This chapter is adapted from [MTC21].

signals [SW05, RKR17, ANR15]. However, in general there can be a combinatorially large number of signals, particularly when the template has a large automorphism group. In such cases, listing or even counting the signals can be computationally intractable. In such situations, it is appealing to have methods that characterize the space of all signals in some way. For example, one might identify the world nodes that participate in at least one signal. Alternatively, one may seek the set of world nodes that correspond to a particular template node in at least one signal. We find that these problems can be feasible, even when it is prohibitive to list or count all of the signals.

Definition 4.1.1 (Multiplex Network)**.** *A* multiplex network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{C})$ *is a set of nodes (frequently called vertices), directed edges between the nodes, labels on the nodes, and channels on the edges. The number of nodes is denoted* n*. Each node* $\mathbf{v} \in \mathcal{V}$ *has a label* $\mathcal{L}(\mathbf{v})$ *belonging to some arbitrary set of labels. There can be any number of edges between each pair of nodes* $(\mathbf{u}, \mathbf{v})$ *in either direction. Each edge belongs to one of the channels* $\mathcal{C}$*. Edges between the same pair of nodes in the same channel with the same direction are indistinguishable. The function* $\mathcal{E} : \mathcal{V} \times \mathcal{V} \to \mathbb{N}^{|\mathcal{C}|}$ *describes the number of edges in each channel between each pair of nodes. In particular,* $\mathcal{E}(\mathbf{u}, \mathbf{v})$ *can be represented as a* $|\mathcal{C}|$*-dimensional vector the* k^{th} *element of which is the number of edges from node* $\mathbf{u}$ *to node* $\mathbf{v}$ *in the* k^{th} *channel. The number of distinguishable edges in* $\mathcal{G}$ *is denoted* $|\mathcal{E}|_0$ *.*

In the remainder of this section, we define several problems related to subgraph matching and discuss existing approaches to solve these problems. In Subsection 4.1.3, we explain our contributions to solving these problems. We expand on the details of our approach in Section 4.2 and Subsections 4.3.1 and 4.3.2. In Section 4.4, we perform several experiments to show that the methods we discuss are successful in solving the problems of interest. We make some concluding remarks in Section 4.5 respectively.

4.1.1 Problem Statements

Given two multiplex networks, a template $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ and a world $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$, we explore the space of all subgraphs of the world that *match* the template. There are several closely related problems with different computational costs. Each of these problems relies on the same concept of *subgraph isomorphism (SI)* as described below.

Definition 4.1.2 (SI: Subgraph Isomorphism)). *An injective function $f : \mathcal{V}_t \to \mathcal{V}_w$ is called a* subgraph isomorphism (SI) *from $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ to $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$ if*

$$\begin{aligned} \mathcal{L}_t(\mathbf{v}) &= \mathcal{L}_w(f(\mathbf{v})) && \forall \mathbf{v} \in \mathcal{V}_t \\ \mathcal{E}_t(\mathbf{u}, \mathbf{v}) &\leq \mathcal{E}_w\left(f(\mathbf{u}), f(\mathbf{v})\right) && \forall \mathbf{u}, \mathbf{v} \in \mathcal{V}_t \times \mathcal{V}_t. \end{aligned}$$

The set of all SIs from $\mathcal{G}_t$ to $\mathcal{G}_w$ is denoted $\mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)$.

In short, an injective function f mapping template nodes to world nodes is an SI if each node $\mathbf{v}$ in the template has the same label as the corresponding node $f(\mathbf{v})$ in the world, and for each pair of nodes $(\mathbf{u}, \mathbf{v})$ in the template, the corresponding pair of nodes $(f(\mathbf{u}), f(\mathbf{v}))$ in the world have *at least as many* edges in each channel and direction as $\mathbf{u}$ and $\mathbf{v}$ have between them. This is sometimes referred to as a "subgraph monomorphism" in the literature. We use the term *induced SI* when there are exactly the same number of edges in each channel and direction. The problems introduced in the remainder of Subsection 4.1.1 are summarized in terms of SIs in Table 4.1.

Nodes in the image of an SI are called *signal nodes*, and the induced subgraph of the image of an SI is called a *signal*. Figure 4.1 highlights several signals in an example problem. Note that there may be more edges between signal nodes than there are between the corresponding template nodes. As an example, in Figure 4.1 there are more edges between **5** and **7** than there are between **B** and **C**.

A typical definition of SI would include a map from template edges to world edges. However, we omit this consideration since we consider edges to be indistinguishable. If edges

Problem	Description
SIP	Check if there are any SIs.
SNSP	Find all the world nodes involved in SIs.
MCSP	Find all pairs $(\mathbf{u}, \mathbf{v})$ where $\mathbf{u} = f(\mathbf{v})$ for some SI f.
SICP	Count the number of SIs.
SMP	Find all the SIs.

Table 4.1: A summary of the various problems defined in Subsection 4.1.1, in increasing order of computation cost.

were considered to be distinct, there would be four SIs in Figure 4.1, all corresponding to signal 2, one for each way to choose the dashed blue edge between template nodes **B** and **C** from among those between world nodes **5** and **7** and the solid green edges between template nodes **A** and **C** from among those between world nodes **4** and **7**.

Definition 4.1.3 (SIP: Subgraph Isomorphism Problem)**.** *Given a template $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ and a world $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$, check if there is at least one SI from the template $\mathcal{G}_t$ to the world $\mathcal{G}_w$. That is,*

$$\textit{check if} \quad \mathcal{F}(\mathcal{G}_t, \mathcal{G}_w) = \emptyset. \tag{SIP}$$

Typically the SIP is solved by exhaustively searching for any SI $f \in \mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)$. If none can be found, then $\mathcal{F}(\mathcal{G}_t, \mathcal{G}_w) = \emptyset$. Though the SIP is NP-complete [GJ79], it can be solved in practice even for some networks with over a billion nodes [SWW12, RKR17]. The challenge of finding all SIs, rather than simply checking whether there are any, is called the *subgraph matching problem (SMP)*.

Definition 4.1.4 (SMP: Subgraph Matching Problem)**.** *Given a template $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ and a world $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$, find all SIs from the template $\mathcal{G}_t$ to the world $\mathcal{G}_w$. That is,*

$$\textit{find all} \quad f \in \mathcal{F}(\mathcal{G}_t, \mathcal{G}_w). \tag{SMP}$$

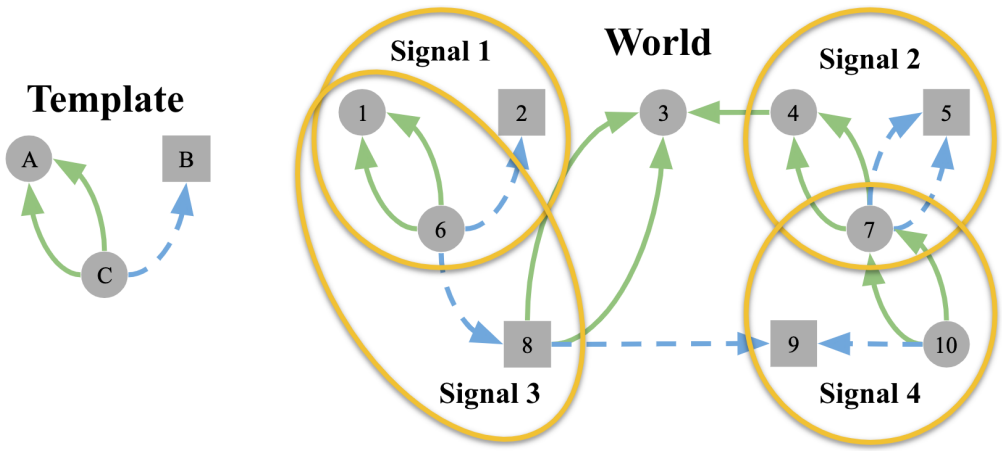


Figure 4.1: In the above networks, the shapes of the nodes corresponds to their labels (circle or square) and the patterns of the edges correspond to their channel (solid green or dashed blue). Given the template and world networks above, there are four signals consisting of the subgraphs of the world induced by $\{\mathbf{1},\mathbf{2},\mathbf{6}\}$, $\{\mathbf{1},\mathbf{6},\mathbf{8}\}$, $\{\mathbf{4},\mathbf{5},\mathbf{7}\}$, and $\{\mathbf{7},\mathbf{9},\mathbf{10}\}$.

The solution to the SMP for Figure 4.1 is the set of SIs $\mathcal{F}(\mathcal{G}_t, \mathcal{G}_w) = \{f_1, f_2, f_3, f_4\}$, where the f_i are described in Table 4.2. Notice that SIs can differ by as little as one node, for example $f_1(\mathbf{v}) = f_3(\mathbf{v})$ for every template node $\mathbf{v}$ except when $\mathbf{v} = \mathbf{B}$. Additionally, some SIs are completely disjoint. For example, $f_2(\mathcal{V}_t) \bigcap f_3(\mathcal{V}_t) = \emptyset$.

Template node $\mathbf{v}$	$f_1(\mathbf{v})$	$f_2(\mathbf{v})$	$f_3(\mathbf{v})$	$f_4(\mathbf{v})$	$\bigcup_i \{f_i(\mathbf{v})\}$
A	**1**	**4**	**1**	**7**	$\{\mathbf{1},\mathbf{4},\mathbf{7}\}$
B	**2**	**5**	**8**	**9**	$\{\mathbf{2},\mathbf{5},\mathbf{8},\mathbf{9}\}$
C	**6**	**7**	**6**	**10**	$\{\mathbf{6},\mathbf{7},\mathbf{10}\}$

Table 4.2: Solutions to SMP and MCSP corresponding to the template and world shown in Figure 4.1.

Since any algorithm for solving the SMP must list all of the SIs $\mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)$, the computation time required, at minimum, must scale with the number of SIs $|\mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)|$. The number of

SIs can be as large as the number of injective maps from $\mathcal{V}_t$ to $\mathcal{V}_w$ which is $O(|\mathcal{V}_w|^{|\mathcal{V}_t|})$. The problems described in the remainder of this section can be used to explore the space of SIs, especially when solving the SMP is computationally prohibitive.

In certain contexts, one may wish to only count the number of SIs from the template to the world. Examples of such contexts include summary statistics of the world network, as in triangle counting [SW05] or motif counting [ADH08]. This is called the *SI counting problem (SICP).*

Definition 4.1.5 (SICP: Subgraph Isomorphism Counting Problem)**.** *Given a template $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ and a world $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$, determine the number of SIs from the template $\mathcal{G}_t$ to the world $\mathcal{G}_w$. That is,*

$$\textit{find} \quad |\mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)|\,. \tag{SICP}$$

Another concise summary of the set of SIs is the set of all signal nodes, i.e. those nodes that are in the image of at least one SI. Finding this set of nodes is the *signal node set problem (SNSP).* The number of signal nodes is bounded from above by the total number of world nodes $|\mathcal{V}_w|$. Though the SNSP lacks the completeness of the full set of SIs, it is much cheaper to calculate.

Definition 4.1.6 (SNSP: Signal Node Set Problem)**.** *Given a template $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ and a world $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$, find all world nodes that belong to at least one signal. That is,*

$$\textit{find} \quad \bigcup_{f \in \mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)} f(\mathcal{V}_t). \tag{SNSP}$$

A compromise between the compactness of the SNSP and the completeness of the SMP is to find the *minimal candidate sets*. For each template node, the minimal candidate set is the smallest set containing all world nodes that correspond to that template node in any signal. This preserves the relation between template nodes and world nodes, but loses some information about compatibility between candidates for different template nodes.

Definition 4.1.7 (MCSP: Minimal Candidate Sets Problem)**.** *Given a template* $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{C})$ *and a world* $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{C})$*, for each template node* $\mathbf{v} \in \mathcal{V}_t$ *find all world nodes which correspond to* $\mathbf{v}$ *in at least one signal. That is,*

$$\textit{find} \quad \bigcup_{f \in \mathcal{F}(\mathcal{G}_t, \mathcal{G}_w)} \{f(\mathbf{v})\} \quad \textit{for each} \quad \mathbf{v} \in \mathcal{V}_t. \tag{MCSP}$$

A naive algorithm for solving the MCSP is to solve the SIP with the added constraint $\mathbf{u} = f(\mathbf{v})$ for each $(\mathbf{u}, \mathbf{v}) \in \mathcal{V}_w \times \mathcal{V}_t$. Thus, solving the MCSP is at most $|\mathcal{V}_t|\,|\mathcal{V}_w|$ times as computationally expensive as the most expensive among those SIPs.

4.1.2 Related Work

Most state of the art algorithms for subgraph matching follow one of three approaches [CFS17, CFS18]: tree-search, constraint-propagation, and graph-indexing. The majority of existing algorithms are specialized to the single-channel case and are not directly applicable to multiplex networks without some adaptation. Also, existing algorithms focus mainly on addressing the SIP, SMP, and SICP, and do not address the MCSP or SNSP directly.

Tree-search approaches keep track of a search state, and navigate the tree of possible search states, backtracking when they reach the end of a branch. Due to the enormity of this tree, to limit computational complexity as much as possible, these approaches refine the search space at each step of the search to avoid unnecessary branches. Examples of tree-search approaches include Ullmann's algorithm [Ull76], VF2 [CFS04] and its variants (VF2Plus [CFV15], VF3 [CFS17, CFS18], VF2++ [JM18]), and for specific graphs, RI/RI-DS [BGP13].

Constraint-propagation approaches view subgraph isomorphism as a constraint satisfaction problem, where variables are assigned values while satisfying a given set of constraints. These approaches keep track of a compatibility matrix which indicates the world nodes that are possible matches for each template node. By repeatedly applying local constraints, this matrix is reduced until only a few possible matches remain. The matrix can then be explored to find

all solutions. Examples of constraint-propagation approaches include McGregor [McG79], nRF+ [LV02], ILF [ZDS10], LAD [Sol10] (and its variants, IncompleteLAD and PathLAD [KMS16]), McCreesh and Prosser (Glasgow) [MP15], and FocusSearch [Ull10].

Graph-indexing approaches seek to retrieve from a database of graphs all graphs that match a given subgraph query. To accelerate searching, they construct indexes for the database, so that future searches will be efficient. These indexes are often based on characteristic substructures of the template. A Cartesian product is then performed on the results of these indexed queries, identifying all possible matches. Next, there is a verification step to check which retrieved graphs fully match the pattern; this typically involves running another subgraph isomorphism algorithm, such as VF2. Examples of graph-indexing approaches include GraphQL [HS08], SPath [ZH10], GADDI [ZLY09], QuickSI [SZL08], TurboISO [HLL13], BoostISO [RW15], CFL-Match [BCL16], and CNI-Match [NS17a]. Our problem does not involve graph databases and we thus do not construct indices. However, several graph-indexing approaches involve filtering techniques which can be used independently of the indices they construct [SL19].

4.1.2.1 Ullmann's Algorithm

Ullmann's algorithm [Ull76] is a backtracking tree-search with refinement. For each template node $\mathbf{v}$, it creates a list of candidate nodes in the world that could correspond to $\mathbf{v}$. Initially, this is simply all nodes with degree greater than or equal to the degree of $\mathbf{v}$. At each step in the tree-search, a candidate node $\mathbf{u}$ is chosen as a match for template node $\mathbf{v}$. To reduce computation time, the remaining candidates are then refined as follows. For every pair of template nodes joined by an edge, their candidates should also be joined by an edge in the world. Any candidate for one of these connected nodes that does not have an edge to any candidate for the other connected node can be removed from consideration. Our topology filter, as detailed in Subsection 4.2.3, is a simple extension of this constraint to the multiplex network case.

4.1.2.2 VF2

VF2 [CFS04] generalizes Ullmann's algorithm to directed graphs and extends refinement with additional semantic feasibility rules. It distinguishes matched nodes (nodes with only one candidate) from other nodes, and ensures that candidates have more matched neighbors and unmatched neighbors than their corresponding template node; in the directed case, these neighbors must have edges with matching directions. It also enforces a matching order where neighbors of previously matched nodes are matched before other nodes.

4.1.2.3 Constraint-Propagation Approaches

In [LV02], the authors classify existing constraints using two categorizations: binary vs. non-binary and forward checking (FC) vs. really forward look ahead (RF). Binary constraints map n variables to m values using an $n \times m$ matrix of binary-valued variables, while non-binary constraints use a vector of length n, whose entries are restricted to m values. The authors propose nRF+, a non-binary really forward lookahead algorithm, with specific look ahead for subgraph isomorphism. They enforce the constraint that if $\mathbf{v}_w$ is a candidate for template node $\mathbf{v}_t$, then the number of $\mathbf{v}_t$'s neighbors must be less than or equal to the number of candidates for those neighbors that are adjacent to $\mathbf{v}_w$.

The work [Sol10] classifies many constraints previously used in different subgraph isomorphism algorithms. The author proposes a new algorithm, LAD, using the constraint that for a match to exist between nodes m and n, there must exist a matching between their neighborhoods subject to the *alldifferent constraint* introduced in [Reg94], which can be identified using the Hopcroft–Karp algorithm [HK73]. The alldifferent constraint requires that the matchings between neighborhoods must be one-to-one.

4.1.2.4 Graph-Indexing Approaches

In GraphQL [HS08], the authors describe a graph query language and provide an algorithm for resolving graph database queries. In particular, they iterate a constraint similar to that of [Sol10], where there must exist a bipartite matching between the neighborhoods of a template node and its candidate. This approach is further expanded upon in SPath [ZH10], where the k-distance neighborhood is also considered.

TurboISO proposes a method for handling permutable template nodes, as well as addressing the order of matching when identifying isomorphisms. It constructs an NEC tree, whose vertices represent groups of permutable template nodes, and perform matching using this tree. It then combines and permutes the candidates for each of these template nodes. For the matching order, it divides the graph into candidate subregions, each of which contains a group of candidates that may take part in a signal. It then prioritizes searching the smaller candidate subregions. We take a similar approach in Subsection 4.3.1, prioritizing nodes with the fewest candidates.

In CFL-Match [BCL16], the template is first decomposed into three types of structures: Core, Forest, and Leaf. To do this, it first constructs a spanning tree of the template. Then, it computes the minimal connected subgraph containing all nontree edges of the template. It then iteratively removes all nodes with degree 1, updating the degree counts after each removal. The remaining nodes form the core.

For each node in the core that is connected to a non-core node, a forest structure is created, consisting of that node and all non-core nodes it is connected to. Finally, the leaf structure consists of chains of removed non-core nodes.

After decomposing the template in this way, each structure is queried independently. In order to postpone the Cartesian product of the results as much as possible, the core is queried first, and the results are then used to restrict queries for each forest structure. Similarly, the leaf queries are restricted by the results of the forest queries.

4.1.2.5 Multiplex Approaches

Though most subgraph isomorphism algorithms in the literature focus on the single-channel networks, there are two algorithms that explicitly solve the multiplex SMP. SuMGrA [IIP16] is a graph-indexing and backtracking tree-search approach and RI [BGP13] can be extended to the multiplex case (MultiRI) [MBF20]. These existing multiplex approaches are designed to solve the SMP. However, the SMP is infeasible when there are too many SIs as in many of the problem instances in Section 4.4. Additionally, the networks considered by SuMGrA and MultiRI differ slightly from Definition 4.1.1. Though they allow multiple edges between nodes, they do not allow multiple edges between a pair of nodes in the same channel. SuMGra also does not allow directed edges.

To extend any existing single-channel algorithms to multiplex networks, one possibility is to use a single-channel approach (e.g., VF2, LAD) to solve the SMP in each channel and take an intersection across all channels. However, this approach is infeasible when there are too many SIs in any channel, even if there are few SIs overall. In Section 4.4, we show examples where the number of SIs is too large to solve the multiplex SMP, let alone the single-channel SMP.

4.1.3 Contributions

We extend existing constraint satisfaction approaches to operate on multiplex networks. We demonstrate experimentally that these approaches allow us to list or count all SIs for world graphs with thousands to hundreds of thousands of nodes and templates with tens to hundreds of nodes.

Due to the underconstrained nature of the templates in some datasets, there are often too many SIs to reasonably list, or sometimes even count. In such situations, we propose that a natural problem to solve in place of the SMP or SICP is the MCSP. On datasets where we can solve the MCSP, we observe that the output of our filters can be a good approximation

to the solution of the MCSP.

We have published our code as an open-source Python package available on GitHub (`https://github.com/jdmoorman/uclasm/tree/master`) [MTC20].

4.2 Filtering

Our algorithms for solving the problems discussed in Subsection 4.1.1 revolve around the use of *filters* to cheaply approximate the solution of the MCSP (Definition 4.1.7). For each template node $\mathbf{v}_t \in \mathcal{V}_t$, we keep track of its *candidates* $D(\mathbf{v}_t) \subseteq \mathcal{V}_w$. Initially, we treat every world node as a candidate for every template node. Each filter enforces a different set of constraints to eliminate candidates that cannot be part of any signal. The filters are applied repeatedly until no further candidates can be eliminated (Algorithm 7), applying the cheaper filters before the more expensive ones.

Algorithm 7 Filtering

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$, list of filters `filters`
`converged` $\leftarrow$ False
while `converged` *is* False
    `converged` $\leftarrow$ True ▷ Stop unless progress is made
    for `filter` $\in$ `filters`
        $D \leftarrow$ `filter`$(\mathcal{G}_t, \mathcal{G}_w, D)$ ▷ Apply the filter
        if $|D(\mathbf{v}_t)|$ decreased for some $\mathbf{v}_t \in \mathcal{V}_t$
            `converged` $\leftarrow$ False ▷ Progress was made
            Break ▷ Restart from the first filter
Output updated candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

In this chapter, in general, we are not searching for induced subgraphs; we do not require equal number of edges to exist between nodes in the template and nodes in the world graph. Instead, we only require the edges between template nodes to be less than or equal to the

number of edges between their corresponding candidates in the world. However, modifying our filters to find induced subgraphs is simple. In Algorithm 9, change $\geq$ to == on Lines 6 and 7, and in Algorithm 11, change $\geq$ to == each time it appears on Line 8. Additionally, all nodes must be considered as "neighbors" of each other, even if they have no edges between them; this affects Line 2 of Algorithm 9 and Lines 4 and 5 of Algorithm 11.

4.2.1 Node Label Filter

In order for a world node $\mathbf{v}_w$ to be a candidate for a template node $\mathbf{v}_t$, the label $\mathcal{L}_w(\mathbf{v}_w)$ must match the label $\mathcal{L}_t(\mathbf{v}_t)$. For example, consider the template and world shown in Figure 4.1 in which the node labels correspond to their shapes. By the label filter, the square world nodes **2**, **5**, **8**, and **9** can be eliminated as candidates for the circular template nodes **A** and **C**, while the circular world nodes **1**, **3**, **4**, **6**, **7**, and **10** can be eliminated as candidates for the square template node **B**. The label filter is run only once, immediately after receiving the template and world.

In some examples, template node labels cannot be specified exactly. For example, in geospatial applications, template node labels may represent broad regions, whereas world node labels may represent exact coordinates. In such applications, it does not make sense to require equality between template node labels and world node labels. Rather, one should require the world node labels to be somehow "compatible" with the template node labels. The notion of compatibility depends on the application. In Subsection 4.4.4.1 we discuss an example application where a world node is compatible with a template node if the coordinates of the world node lie within the region of the template node.

4.2.2 Node-level Statistics Filter

The idea behind the node-level statistics filter (Algorithm 8) is intuitive: for a world node $\mathbf{v}_w$ to be a candidate for a template node $\mathbf{v}_t$, certain statistical properties of $\mathbf{v}_w$ should not be less than those of $\mathbf{v}_t$. The idea of the node-level statistics filter has been applied to simpler

settings in the related literature [Sol10, MBF20]. Any statistic that is non-decreasing as nodes and/or edges are added to a graph can be used as part of the filter. The statistics that are applied in our filter include in/out-degree, number of in/out-neighbors, number of reciprocated edges, and number of self-edges. Each of these statistics is used in each channel in the networks.

Algorithm 8 Node-level Statistics Filter

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$
for `statistic` $\in$ [in-degree, out-degree, ...]
    for template or world node $\mathbf{v} \in \mathcal{V}_t \cup \mathcal{V}_w$
        Compute `statistic`($\mathbf{v}$)
    for template node $\mathbf{v}_t \in \mathcal{V}_t$
        for candidate $\mathbf{v}_w \in D(\mathbf{v}_t)$
            if `statistic`($\mathbf{v}_w$) < `statistic`($\mathbf{v}_t$)
                Remove $\mathbf{v}_w$ from $D(\mathbf{v}_t)$
Output updated candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

Other more complex statistics can be used channel-wise, such as number of triangles, number of nodes within k steps, and number of paths of length ℓ. Statistics combining information from multiple channels can also be used, such as the number of in-neighbors in channel a that are also out-neighbors in channel b. We use only statistics the computational cost scales linearly in the number of distinguishable edges (edges whose source, destination, direction, and channel are distinct).

As an example, consider applying an in/out-degree filter to the problem in Figure 4.2. The in-degree and out-degree of each template and world node are listed in Table 4.3. Node **A** has one outgoing edge and one incoming edge, thus nodes **2** and **5** can be ruled out as candidates since node **2** has no outgoing edges and node **5** has no incoming edges. Similarly, node **B** has one incoming edge, so node **5** can be ruled out as a candidate since it does not have any incoming edges. Finally, node **C** has two outgoing edges and one incoming edge, so

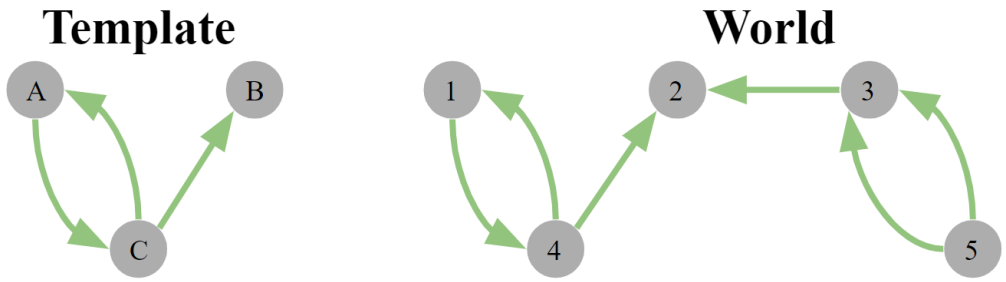


Figure 4.2: In the network shown, there is only one valid signal for the template: **1** for **A**, **2** for **B**, and **4** for **C**.

	Template			World				
	A	**B**	**C**	**1**	**2**	**3**	**4**	**5**
in-degree	1	1	1	1	2	2	1	0
out-degree	1	0	2	1	0	1	2	2

Table 4.3: In/out-degree for nodes in the template and world shown in Figure 4.2

all world nodes except **4** are eliminated as candidates since **4** is the only world node with at least two outgoing edges *and* at least one incoming edge. The candidates for each template node after applying the node-level statistics filter are summarized in the "Statistics" column of Table 4.4.

The node-level statistics filter is the second cheapest filter to apply, after the node label filter, and is most effective when some template nodes have statistic values that are uncommon in the world. Narrowing down the candidates for even one template node can help the other filters to refine the candidates for the remaining template nodes.

4.2.3 Topology Filter

In the topology filter, we enforce the constraint proposed by [Ull76], extended to multiplex networks. The original constraint, denoted as *AC(Edges)* [Sol10], is that if $\mathbf{v}_w$ is a candidate

	Filters run		
	statistics	statistics, topology	statistics, topology, repeated-sets
Candidates for **A**	**1**, **3**, **4**	**1**	**1**
Candidates for **B**	**1**, **2**, **3**, **4**	**1**, **2**	**2**
Candidates for **C**	**4**	**4**	**4**

Table 4.4: Candidates per template node for the problem shown in Figure 4.2 after various filters have been applied.

for template node $\mathbf{v}_t$, then for every template node $\mathbf{u}_t$ that neighbors $\mathbf{v}_t$, there must exist a candidate $\mathbf{u}_w$ for $\mathbf{u}_t$ that neighbors $\mathbf{v}_w$. The natural extension to multiplex networks is that if $\mathbf{v}_w$ is a candidate for template node $\mathbf{v}_t$, then for every template node $\mathbf{u}_t$ that neighbors $\mathbf{v}_t$, there must exist a candidate $\mathbf{u}_w$ for $\mathbf{u}_t$ that has as many edges in each channel and direction between $\mathbf{v}_w$ and $\mathbf{u}_w$ as there are between $\mathbf{v}_t$ and $\mathbf{u}_t$.

To clarify the concept, consider applying the topology filter to the problem in Figure 4.2 after applying the node-level statistics filter. Since node **4** is the only candidate for node **C**, nodes **3** and **4** are eliminated as candidates for both nodes **A** and **B** because they do not have neighbors that are candidates for node **C**. These results are shown in Table 4.4.

In terms of computational complexity, the topology filter scales linearly with the number of distinguishable template edges, and with the number of world nodes squared. In practice, by using sparse matrices, the second factor can be reduced to the number of distinguishable world edges leading to a computational complexity of $O(|\mathcal{E}_t|_0 \, |\mathcal{E}_w|_0)$.

The topology filter is particularly useful when one or more template nodes have few candidates remaining after applying other filters. In such situations, it can significantly reduce the candidates for neighboring template nodes.

Algorithm 9 Topology Filter

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$
for neighboring template nodes $\mathbf{v}_t$ and $\mathbf{u}_t$
    for candidate $\mathbf{v}_w \in D(\mathbf{v}_t)$
        `nbr_cand_found` $\leftarrow$ False
        for candidate $\mathbf{u}_w \in D(\mathbf{u}_t)$
            `enough_out` $\leftarrow \mathcal{E}_w(\mathbf{v}_w, \mathbf{u}_w) \geq \mathcal{E}_t(\mathbf{v}_t, \mathbf{u}_t)$
            `enough_in` $\leftarrow \mathcal{E}_w(\mathbf{u}_w, \mathbf{v}_w) \geq \mathcal{E}_t(\mathbf{u}_t, \mathbf{v}_t)$
            if `enough_out` *and* `enough_in`
                `nbr_cand_found` $\leftarrow$ True
                break
        if `nbr_cand_found` *is not* True
            Remove $\mathbf{v}_w$ from $D(\mathbf{v}_t)$
Output updated candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

4.2.4 Repeated-Sets Filter

Here, we apply another constraint, *GAC(AllDiff)* [Hoe, Sol10]: generalized arc consistency (also known as hyper-arc consistency) for the alldifferent constraint. *GAC(AllDiff)* requires that for a world node $\mathbf{v}_w$ to be a candidate for template node $\mathbf{v}_t$, there must exist some injective mapping from the template nodes to their candidates under which $\mathbf{v}_t$ is mapped to $\mathbf{v}_w$. To enforce *GAC(AllDiff)*, we identify sets of template nodes $\mathcal{T} \subseteq \mathcal{V}_t$ where the union of their candidates $\bigcup_{\mathbf{v}_t \in \mathcal{T}} D(\mathbf{v}_t)$ has the same cardinality as $\mathcal{T}$. These are known as *tight sets* [Hoe]. Candidates for template nodes in a tight set cannot be candidates for any nodes outside of the tight set, since this would violate *GAC(AllDiff)*.

Standard algorithms for enforcing *GAC(AllDiff)* run in $O\Big(\sqrt{|\mathcal{V}_t| + |\mathcal{V}_w|} \sum_{\mathbf{v}_t \in \mathcal{V}_t} |D(\mathbf{v}_t)|\Big)$ time complexity [Reg94, GMN08]. Some improvements and modifications to the standard algorithms have been explored to mixed benefit [GMN08]. In the repeated-sets filter (Algorithm 10), we enforce *GAC(AllDiff)* by directly considering unions of candidate sets to

find tight sets. Though the number of candidate set unions grows exponentially with the number of template nodes [Hoe], we restrict this growth by not considering unions with more nodes than there are template nodes. We also keep track of template nodes that belong to tight sets and do not use them in unions. In practice, templates are often hundreds of nodes or smaller, and we don't observe the worst case exponential scaling. In terms of the world graph, naive maximum cardinality matching-based algorithms scale as $O(|\mathcal{V}_w|^{3/2})$, whereas Algorithm 10 scales linearly with $|\mathcal{V}_w|$.

To illustrate the application of the repeated-sets filter, consider the example in Figure 4.2. Using the node-level statistics and topology filters, the candidates for the template nodes were narrowed down to those in Table 4.4. Since $\mathbf{A}$ has only one candidate, $\{\mathbf{A}\}$ is a tight set and $\mathbf{1}$ can be removed as a candidate for the remaining template nodes. This leaves only one candidate for each template node, exactly corresponding to the only signal in Figure 4.2.

The repeated-sets filter is most important when some template nodes have only one candidate, since any template node with only one candidate forms a tight set. It is also useful for templates such as those in Subsections 4.4.5.1 and 4.4.5.5 that contain nodes that are indistinguishable.

4.2.5 Neighborhood Filter

In this filter, we extend the *local alldifferent (LAD)* constraint introduced in [Sol10] to the multiplex network case. In the undirected single-channel graph context, the LAD constraint ensures that for a world node $\mathbf{v}_w$ to be a candidate for a template node $\mathbf{v}_t$, there must be some injective mapping f_ℓ from the neighbors of $\mathbf{v}_t$ to their candidates under which $f_\ell(\mathbf{u}_t)$ is a neighbor of $\mathbf{v}_w$ for each $\mathbf{u}_t$. We extend this to the multiplex network context by requiring not just that $f_\ell(\mathbf{u}_t)$ neighbors of $\mathbf{v}_w$, but also that there are enough edges in each channel and direction

$$\begin{aligned} &\mathcal{E}_w(\mathbf{v}_w, \mathbf{u}_w) \geq \mathcal{E}_t(\mathbf{v}_t, \mathbf{u}_t) \\ \text{and} \quad &\mathcal{E}_w(\mathbf{u}_w, \mathbf{v}_w) \geq \mathcal{E}_t(\mathbf{u}_t, \mathbf{v}_t) \end{aligned} \tag{4.1}$$

Algorithm 10 Repeated-Sets Filter

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

`unions` $\leftarrow$ `EmptyMap()`

▷ Map sets of world nodes $\mathcal{U}_w$ to sets of template nodes $\mathcal{U}_t$ for which $\mathcal{U}_w = \bigcup_{\mathbf{v}_t \in \mathcal{U}_t} D(\mathbf{v}_t)$. Any $\mathcal{U}_t$ with $|$`unions`$[\mathcal{U}_t]| == |\mathcal{U}_t|$ is a tight set.

$\mathcal{T} \leftarrow \emptyset$ ▷ Nodes known to belong to tight sets.

`todo` $\leftarrow \{\mathbf{v}_t \in \mathcal{V}_t : |D(\mathbf{v}_t)| < |\mathcal{V}_t|\}$ ▷ Template nodes with too many candidates cannot belong to a tight set.

while $|$`todo`$| > 0$

  $\mathbf{v}_t \leftarrow$ element of `todo` with fewest candidates $D(\mathbf{v}_t)$

  `todo` $\leftarrow$ `todo` $\setminus \{\mathbf{v}_t\}$

  for $(\mathcal{U}_w, \mathcal{U}_t) \in$ `unions`

    $\mathcal{U}_w \leftarrow \mathcal{U}_w \cup D(\mathbf{v}_t)$

    if $|\mathcal{U}_w| < |\mathcal{V}_t|$

      $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{\mathbf{v}_t\}$

      if $|\mathcal{U}_t| == |\mathcal{U}_w|$ ▷ $\mathcal{U}_t$ is a tight set.

        $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{U}_t$

        $D(\mathbf{u}_t) \leftarrow D(\mathbf{u}_t) \setminus \mathcal{U}_w$ for $\mathbf{u}_t \in \mathcal{V}_t \setminus \mathcal{U}_t$

      else

        `unions`$[\mathcal{U}_w] \leftarrow \mathcal{U}_t$

  if $D(\mathbf{v}_t) \notin$ `unions`

    if $|D(\mathbf{v}_t)| == 1$ ▷ $\{\mathbf{v}_t\}$ is a tight set.

      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{v}_t\}$

      $D(\mathbf{u}_t) \leftarrow D(\mathbf{u}_t) \setminus D(\mathbf{v}_t)$ for $\mathbf{u}_t \in \mathcal{V}_t \setminus \{\mathbf{v}_t\}$

    else

      `unions`$[D(\mathbf{v}_t)] \leftarrow \{\mathbf{v}_t\}$

  `todo` $\leftarrow \{\mathbf{v}_t \in$ `todo` $: |D(\mathbf{v}_t)| < |\mathcal{V}_t \setminus \mathcal{T}|\}$

Output updated candidate set $D(\mathbf{v}_t)$ for $\mathbf{v}_t \in \mathcal{V}_t$

where $\mathbf{u}_w = f_\ell(\mathbf{u}_t)$. In the neighborhood filter (Algorithm 11), we enforce the multiplex LAD constraint by searching for such a mapping f_ℓ. If none exists, we eliminate $\mathbf{v}_w$ as a candidate for $\mathbf{v}_t$.

We now transform the search for a mapping f_ℓ into a matching problem on a bipartite graph $\mathcal{B}$. Let $\mathcal{N}_{\mathbf{v}_t}$ and $\mathcal{N}_{\mathbf{v}_w}$ denote the neighborhoods of $\mathbf{v}_t$ and $\mathbf{v}_w$ respectively. Define the undirected bipartite graph $\mathcal{B}$ with parts $\mathcal{N}_{\mathbf{v}_t}$ and $\mathcal{N}_{\mathbf{v}_w}$ to have an edge between nodes $\mathbf{u}_t \in \mathcal{N}_{\mathbf{v}_t}$ and $\mathbf{u}_w \in \mathcal{N}_{\mathbf{v}_w}$ whenever $\mathbf{u}_w$ is a candidate for $\mathbf{u}_t$ and Equation (4.1) holds. A matching on $\mathcal{B}$ is a subset of its edges where no two edges share a node. A mapping f_ℓ is equivalent to a matching on $\mathcal{B}$ of size $|\mathcal{N}_{\mathbf{v}_t}|$ where each edge $(\mathbf{u}_t, \mathbf{u}_w)$ in the matching corresponds to $f_\ell(\mathbf{u}_t) = \mathbf{u}_w$. The Hopcroft–Karp algorithm [HK73] can be used to find the maximum cardinality matching on $\mathcal{B}$ in $O(\sqrt{|\mathcal{N}_{\mathbf{v}_t}| + |\mathcal{N}_{\mathbf{v}_w}|}\,|\mathcal{E}_\mathcal{B}|)$ time, where $\mathcal{E}_\mathcal{B}$ denotes the set of edges of $\mathcal{B}$.

In the example from earlier, in Figure 4.2, the node-level statistics, topology, and repeated-sets filters were sufficient to narrow down the candidates until they solve the MCSP (Definition 4.1.7). In Figure 4.3, we give an example where those filters are not sufficient. The resulting candidates before and after neighborhood filter are given in Table 4.5. Before the neighborhood filter is applied, node **4** remains a candidate for node **C** because node **3** is a candidate for its neighbors **A** and **B**. The biadjacency matrix of $\mathcal{B}$ corresponding to $\mathbf{v}_t = \mathbf{C}$ and $\mathbf{v}_w = \mathbf{4}$ is shown in Table 4.6. Each row of the biadjacency matrix corresponds to a neighbor of $\mathbf{v}_t$ and each column corresponds to a neighbor of $\mathbf{v}_w$. Entries correspond to the conditions on Line 8 of Algorithm 11. Since the maximum cardinality matching on $\mathcal{B}$ has only two elements, the neighborhood filter is able to eliminate node **4** from the candidates of node **C**.

4.2.6 Elimination Filter

The elimination filter (Algorithm 12) attempts to eliminate candidates by identifying any contradictions that would result from them being assigned. For each template node-candidate

Algorithm 11 Neighborhood Filter

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

for template node $\mathbf{v}_t \in \mathcal{V}_t$

  for candidate $\mathbf{v}_w \in D(\mathbf{v}_t)$

    $\mathcal{N}_{\mathbf{v}_t} \leftarrow$ `Neighborhood`$(\mathbf{v}_t)$

    $\mathcal{N}_{\mathbf{v}_w} \leftarrow$ `Neighborhood`$(\mathbf{v}_w)$

    $\mathcal{B} \leftarrow$ `EmptyBipartiteGraph`$(\mathcal{N}_{\mathbf{v}_t}, \mathcal{N}_{\mathbf{v}_w})$

    for $(\mathbf{u}_t, \mathbf{u}_w) \in \mathcal{N}_{\mathbf{v}_t} \times \mathcal{N}_{\mathbf{v}_w}$

      if $\begin{cases} \mathbf{u}_w \in D(\mathbf{u}_t) \\ \textit{and } \mathcal{E}_w(\mathbf{v}_w, \mathbf{u}_w) \geq \mathcal{E}_t(\mathbf{v}_t, \mathbf{u}_t) \\ \textit{and } \mathcal{E}_w(\mathbf{u}_w, \mathbf{v}_w) \geq \mathcal{E}_t(\mathbf{u}_t, \mathbf{v}_t) \end{cases}$

        Add an edge between $\mathbf{u}_t$ and $\mathbf{u}_w$ in $\mathcal{B}$

    `max_match` $=$ `MaxCardinalityMatching`$(\mathcal{B})$

    if $|$`max_match`$| < |\mathcal{N}_{\mathbf{v}_t}|$

      Remove $\mathbf{v}_w$ from $D(\mathbf{v}_t)$

Output updated candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

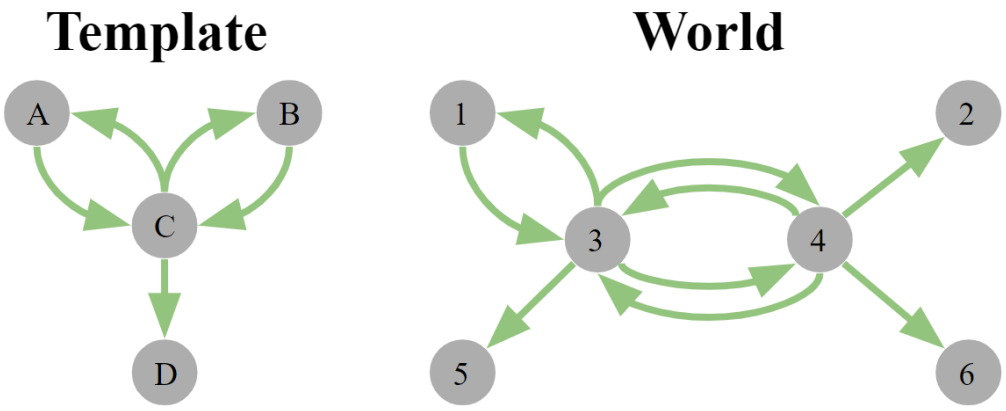


Figure 4.3: An example template and world for which the neighborhood filter plays a role in eliminating candidates.

pair ($\mathbf{v}_t$,$\mathbf{v}_w$), we do a one step lookahead. We assign $\mathbf{v}_w$ to $\mathbf{v}_t$ and iterate over all other filters until convergence. If this results in one or more template nodes having no candidates, then $\mathbf{v}_t$ cannot be mapped to $\mathbf{v}_w$ and we eliminate $\mathbf{v}_w$ as a candidate for $\mathbf{v}_t$.

Algorithm 12 Elimination Filter

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$
$S \leftarrow \texttt{copy}(D)$ ▷ Save the candidate sets
for template node $\mathbf{v}_t \in \mathcal{V}_t$
    for candidate $\mathbf{v}_w \in D(\mathbf{v}_t)$
        $D(\mathbf{v}_t) \leftarrow \{\mathbf{v}_w\}$ ▷ Assign $\mathbf{v}_w$ to $\mathbf{v}_t$
        Iterate all filters to convergence
        if $D(\mathbf{u}_t)$ is empty for any $\mathbf{u}_t \in \mathcal{V}_t$
            Remove $\mathbf{v}_w$ from $D(\mathbf{v}_t)$ in S
        Reset candidate sets to S
Output updated candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

As the elimination filter is a very expensive operation, scaling with the number of remaining candidates, we restrict its use until all other filters have converged and no further candidates can be removed by other means. In certain contexts, the elimination filter can be impractical

	Filters run	
	statistics, topology, repeated-sets	statistics, topology, repeated-sets, neighborhood
Candidates for **A**	**1**, **3**, **4**	**1**, **4**
Candidates for **B**	**1**, **3**, **4**	**1**, **4**
Candidates for **C**	**3**, **4**	**3**
Candidates for **D**	**2**, **5**, **6**	**5**

Table 4.5: Candidates per template node for the problem shown in Figure 4.3 after various filters have been applied.

to use. However, in others, it can greatly reduce the number of candidates. A simple example where elimination filter proves useful can be seen in Figure 4.4, where we have three cycle graphs consisting of 3, 4 and 5 nodes respectively. We also assume that each edge is bidirectional and there is only one channel.

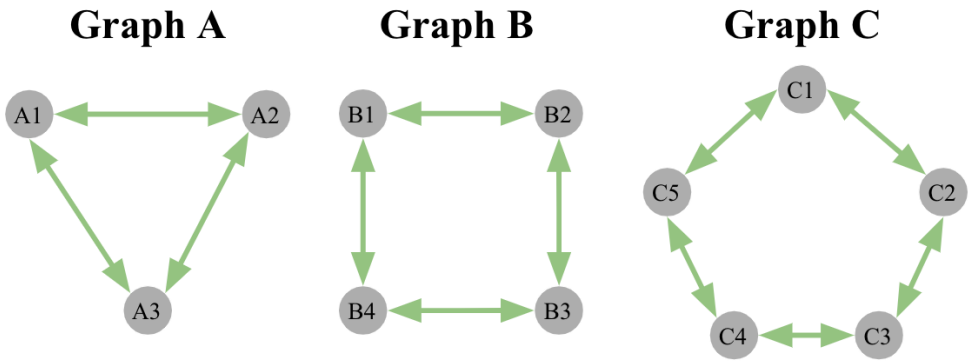


Figure 4.4: Subgraph matching problems consisting of Graph A, B and C.

Suppose we use Graph A as the template graph and Graph B as the world graph (see Figure 4.4). After applying the node-level statistics, topology, and neighborhood filters, every

$\mathcal{N}_C$ \ $\mathcal{N}_4$	**2**	**3**	**6**
A	0	1	0
B	0	1	0
D	1	0	1

Table 4.6: Biadjacency matrix of $\mathcal{B}$ used in the matching problem between the neighborhoods of template node **C** and world node **4**.

node in Graph B remains a candidate of every template node in Graph A. By including elimination filter, we are able to correctly tell that there is no valid signal for this problem. A more difficult problem arises when we use Graph B as the template graph and Graph C as the world graph. In this case, the node-level statistics, topology, and neighborhood filters do not get rid of any invalid candidates. Only after additionally applying the elimination filter can we conclude that there is no valid signal existing in Graph C.

In practice, the elimination filter that iterates over node-level statistics, topology, and repeated-sets filters is often sufficient, since iterating over neighborhood filters is expensive. However, in situations like the second example posed above, we might need to apply all of the existing filters to determine the solution. In particular, the elimination filter is useful in many synthetic and real-world datasets, especially in some of the examples discussed in Subsection 4.4.5.

4.3 Solving the Problems

We present the details of how to solve the SICP and MCSP using the filters from Section 4.2 as a subroutine. The SIP and SMP can be solved with similar methods to SICP, and the SNSP can be solved with a similar method to MCSP.

4.3.1 Isomorphism Counting

After applying the filters described in Section 4.2, some template nodes may have exactly one candidate. Template nodes that still have multiple candidates we refer to as *unspecified* nodes. When an edge exists between two unspecified nodes, we have to enforce that a corresponding edge exists between the two candidates we choose for them. As this makes counting isomorphisms computationally complex, we start by finding a set of unspecified nodes that, if specified, would cause the remaining unspecified nodes to have no edges between them. This set is called a *node cover*, and the smallest such set is called the *minimal node cover*.

For example, in Figure 4.2, if all three template nodes have multiple candidates, the minimal node cover would be $\{\mathbf{C}\}$. Since the minimal node cover NP-hard to compute in general, we settle for a small node cover [BE85].

Next, we iterate through all possible choices of candidates for nodes in the node cover. For each choice, we reapply the topology and repeated-sets filters so we can be sure that any remaining candidates belong to signals. Since the remaining unspecified nodes have no edges between them, it is much simpler to count the ways to choose their candidates. The only constraint is that the same candidate cannot be chosen for more than one node, which is simply the alldifferent constraint satisfaction problem [Reg94].

Simple modifications can be made in order to solve the SIP and SMP. For the SIP, the algorithm exits and returns true once a solution is found, i.e. once `CountAllDiff` (Line 4 of Algorithm 13) returns a nonzero result. For the SMP, replace `CountAllDiff` with an algorithm that returns a list of all solutions to the alldifferent problem, and instead of adding the counts together, combine the two lists of solutions.

Algorithm 13 Isomorphism Counting

1: **function** IsoCount

2: **Input** template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$, unspecified node cover $\mathcal{NC}$

3: **if** $\mathcal{NC} = \emptyset$

4: $\mathtt{N_{isos}} \leftarrow \mathtt{CountAllDiff}(D)$

5: **else**

6: $\mathtt{N_{isos}} \leftarrow 0$

7: $\mathbf{v}_t \leftarrow \mathcal{NC}.\mathtt{pop}()$

8: $S \leftarrow \mathtt{copy}(D)$ ▷ Save the candidate sets

9: **for** world node $\mathbf{v}_w \in D(\mathbf{v}_t)$

10: $D(\mathbf{v}_t) \leftarrow \{\mathbf{v}_w\}$ ▷ Assign $\mathbf{v}_w$ to $\mathbf{v}_t$

11: Apply filters

12: $\mathtt{N_{isos}} \leftarrow \mathtt{N_{isos}} + \mathtt{IsoCount}(\mathcal{G}_t, \mathcal{G}_w, D, \mathcal{NC})$

13: Reset candidate sets to S

14: **Output** number of isomorphisms $\mathtt{N_{isos}}$

4.3.2 Validation

A simpler problem than listing or counting all isomorphisms (SMP or SICP) is the minimal candidate sets problem (MCSP) of Definition 4.1.7. Here, we refer to the procedure for solving the MCSP as *validation*, in which we seek to identify all template node-candidate pairs that participate in at least one isomorphism. An algorithm for validation is as follows (Algorithm 14). First, pick an unvalidated pair of a template node and its candidate. Next, make assignments until an isomorphism is found, running filters after each step and backtracking as needed. If all possible assignments fail, then the pair is invalid; remove the pair from the set of possible candidates. Otherwise, validate every assignment made as part of the isomorphism, including the initial one; these pairs are validated as taking part in at least one isomorphism. Repeat this process until all pairs are validated or removed from candidates. This ensures that every candidate-template node pairing remaining after validation is valid i.e. participates in at least one signal. This is because in order to pass validation, it must participate in at least one isomorphism that was found, and in order to fail, there must not exist an isomorphism containing it.

An optimization for Algorithm 14 when using the node cover approach in Subsection 4.3.1 is that if the node cover has been assigned, the topology and repeated-sets filters have been run to convergence, and the problem has thus been reduced to an alldifferent problem. Then, instead of attempting to find a single solution to the alldifferent problem, all possible candidate pairs corresponding to the alldifferent problem can be simultaneously validated. This is because all remaining candidate pairs passed the repeated-sets filter, which enforces *GAC(AllDiff)*: thus, there must exist a solution for each pair, and thus an isomorphism since edges were previously enforced by the topology filter.

This algorithm can also be modified to solve the SNSP. If, instead of adding the tuple $(\mathbf{u}_t, f(\mathbf{u}_t))$ to `validated`, only the element $f(\mathbf{u}_t)$ is added, then the algorithm will validate only which world nodes participate, without respect to which template nodes they are candidates for.

Algorithm 14 Validation

Input template $\mathcal{G}_t$, world $\mathcal{G}_w$, candidate set $D(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$
$D'(\mathbf{v}_t) \leftarrow \emptyset$ for each $\mathbf{v}_t \in \mathcal{V}_t$ ▷ Valid candidates
for template node $\mathbf{v}_t \in \mathcal{V}_t$
    for candidate $\mathbf{v}_w \in D(\mathbf{v}_t) \setminus D'(\mathbf{v}_t)$
        $S \leftarrow \texttt{copy}(D)$
        $S(\mathbf{v}_t) \leftarrow \{\mathbf{v}_w\}$ ▷ Assign $\mathbf{v}_w$ to $\mathbf{v}_t$
        Attempt to find an SI f using S ▷ Solve the SIP
        if isomorphism found
            $D'(\mathbf{u}_t) \leftarrow D'(\mathbf{u}_t) \cup \{f(\mathbf{u}_t)\}$ for each $\mathbf{u}_t \in \mathcal{V}_t$
        else
            Remove $\mathbf{v}_w$ from $D(\mathbf{v}_t)$
Output minimal candidate set $D'(\mathbf{v}_t)$ for each $\mathbf{v}_t \in \mathcal{V}_t$

4.4 Experiments

Unless otherwise specified, the experiments were run on an HP Z8 G4 Workstation with two 12-core 6,136 3.0 2,666MHz CPUs, using version 0.2.0 of the python code available on GitHub (`https://github.com/jdmoorman/uclasm/tree/v0.2.0`) [MTC20]. The node-level statistics, topology, and repeated-sets filters run in under a minute for all datasets and under a second for most. The elimination filter takes longer, between minutes and hours, depending on the dataset. Validation, if used, can also take hours or longer. The combinatorial complexity of the solution space is largely responsible for the long time needed for the elimination, counting and validation.

4.4.1 Sudoku

The puzzle game of Sudoku involves a 9×9 grid that is partitioned into nine 3×3 *blocks*. Each *cell* of the grid must be filled with a digit 1–9 so that each row, column, and block

	1			8			9	
6		4	9		2	5		3
	5	6	2		8	9	3	
2								1
	3	8	1		7	6	2	
3		1	8		6	2		5
	9			7			8	

Figure 4.5: Sudoku grid with initial clues.

contains each digit 1–9 exactly once. A Sudoku puzzle begins with some of the cells pre-filled with digits so that there is exactly one correct solution (Figure 4.5).

One algorithm for solving Sudoku is Donald Knuth's dancing links algorithm [Knu00], which models Sudoku as an exact cover problem. This is a backtracking tree-search with refinement on a sparse Boolean matrix, using doubly linked lists for efficiency. Sudoku can also be modeled as a constraint satisfaction problem [Sim05]; see e.g. Peter Norvig's constraint-propagation algorithm [Norb]. Here, the two constraints propagated are that cells in the same row must contain different digits and that every cell must have a digit. After these constraints have been iterated, the algorithm uses a depth first backtracking search to find the solution.

The constraint satisfaction problem to solve Sudoku has much in common with subgraph isormorphism algorithms; here we show that Sudoku can be written in multiple ways as a subgraph isomorphism problem. We chose this example to test our algorithms primarily because constraint-propagation is considered state of the art for Sudoku. Here, our goal is to validate the reasonable use of our code on this problem, rather than trying to beat the best Sudoku solvers.

4.4.1.1 9×9 Representation

Sudoku is equivalent to a single-channel subgraph isomorphism problem using the following correspondence: consider the cells as nodes of the template, two cells in the template are linked if they are in the same row, column, or 3×3 block. We consider the world as a set of 81 nodes, each corresponding to a digit. Each digit is locked to a particular 3×3 block, leading to nine sets of nine digits each, having values 1–9. And here, two nodes are linked if they do *not* have the same value. This graph has the same number of nodes, but more edges than the template, so this is a subgraph isomorphism problem. To represent known digits (to start the puzzle), we can restrict the initial candidates. This can be done simply: if a cell is filled in with a value, we identify which 3×3 block the cell is in, find the digit in the world that corresponds to that block and has the matching value, and say that the only candidate for the cell is that digit. The Sudoku puzzle is thus a semi-supervised version of the subgraph graph isomorphism problem and is designed to have a unique solution, unlike the unsupervised examples in our other experiments.

4.4.1.2 $9 \times 9 \times 3$ Representation

One can also assign different edge types (i.e. row, column or block correspondences) to different channels, along with another channel that identifies which cells are the same. The template has three nodes for each cell, a row-node, a column-node, and a block-node. All three are linked by edges in a special "same-cell" channel. Otherwise, the three channels are completely separate: row-nodes are only linked to other row-nodes in the same row, column-nodes to column-nodes, etc. The world is then three channels of nine digits each, so there are once again the same number of world nodes as template nodes. In each channel, sets of nine digits are identified as being in the same row/column/block by edges. In these channels, there are the same number of edges as in the template. However, for the "same-cell" channel, we add an edge between any two nodes that could represent the same cell. Since each 3×3 block only intersects three rows and three columns, these edges are restricted.

Similar to before, we add an additional initial restriction that a row-node in the template can only have the nine row-digits in the world corresponding to its row as candidates, and the same for columns and blocks.

4.4.1.3 Results

To test our filters, we use the sets of Sudoku puzzles obtained from Peter Norvig's GitHub [Nora], including those from Project Euler problem 96 [Pro]. Though our code is not specialized for the task, it still solves all Sudoku examples in the dataset. The time taken to solve each Sudoku example is plotted in Figure 4.6. Overall, it appears that the 9×9 representation works best on the easier puzzles, while the $9 \times 9 \times 3$ representation is faster on harder puzzles. On the hardest puzzles, the 9×9 representation is faster on average; this is primarily due to outliers for which the $9 \times 9 \times 3$ representation takes much longer.

4.4.2 Multiplex Erdős–Rényi

To test the scalability of our algorithm, we perform experiments on randomly generated multiplex Erdős-Rényi networks, where each edge in each channel has the same probability p of occurring. For the templates, $p_t = 0.5$. For the worlds, the probability

$$p_w = 1 - \frac{1 - \sqrt[|\mathcal{C}|]{1 - p_t + p_t^2}}{p_t}$$

is scaled up as the number of channels increases to maintain a fixed probability of length $|\mathcal{C}|$ edge vectors matching

$$\mathbb{P}(\mathcal{E}_t(\mathbf{u}_t, \mathbf{v}_t) \leq \mathcal{E}_w(\mathbf{u}_w, \mathbf{v}_w)) = 0.75$$

for all template nodes $\mathbf{u}_t$, $\mathbf{v}_t$ and world nodes $\mathbf{u}_w$, $\mathbf{v}_w$. Templates are generated with 10 nodes, while worlds are generated with sizes ranging from 10 nodes to 300 nodes. We cap the algorithm runtime at 10,000 seconds per instance.

To measure the difficulty of each instance, we count the number of search iterations (recursive calls to `IsoCount`) taken by Algorithm 13 to solve the SIP and SICP. For the

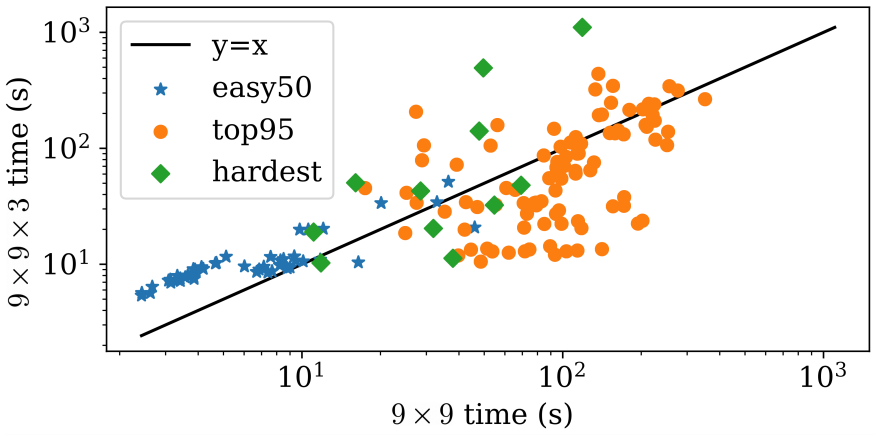


Figure 4.6: Solving Sudoku puzzles as special case of a multiplex subgraph isomorphism problem using the 9×9 and $9 \times 9 \times 3$ representations. Scatter plot of solution times (seconds) on the Sudoku puzzles from Peter Norvig's GitHub [Nora], including those from Project Euler problem 96 [Pro]. A black line is drawn where the representations take an equal amount of time to aid in comparison. Mean solution time for the 9×9 representation is 8.33 seconds on the 50 easy puzzles, 116.7 seconds on the top 95 puzzles, and 43.4 seconds on the hardest puzzles. Mean solution time for the $9 \times 9 \times 3$ representation is 11.24 seconds on the 50 easy puzzles, 95.4 seconds on the top 95 puzzles, and 179.4 seconds on the hardest puzzles.

SIP, the counting is terminated after a single isomorphism is found, or once the entire search space has been checked and no isomorphisms are found. We observe similar difficulty scaling for one, two, and three channel instances (Figure 4.7).

For the SIP, there is an initial sharp rise in search iterations with a peak around 100 iterations, followed by a decline back to a constant level of 10 iterations. This pattern is partially due to the SIP being easier when SIs are either extremely likely or extremely unlikely [MPS18]. When SIs are unlikely, as when the world is near the size of the template, filtering often rules out all SIs before reaching the bottom of the search tree. When SIs are common, as when the world is large (e.g. over 100 nodes), every branch of the search tree is likely to have one, so the number of search iterations converges to the number of template nodes, 10.

For the SICP, the mean number of search iterations scales monotonically with the number

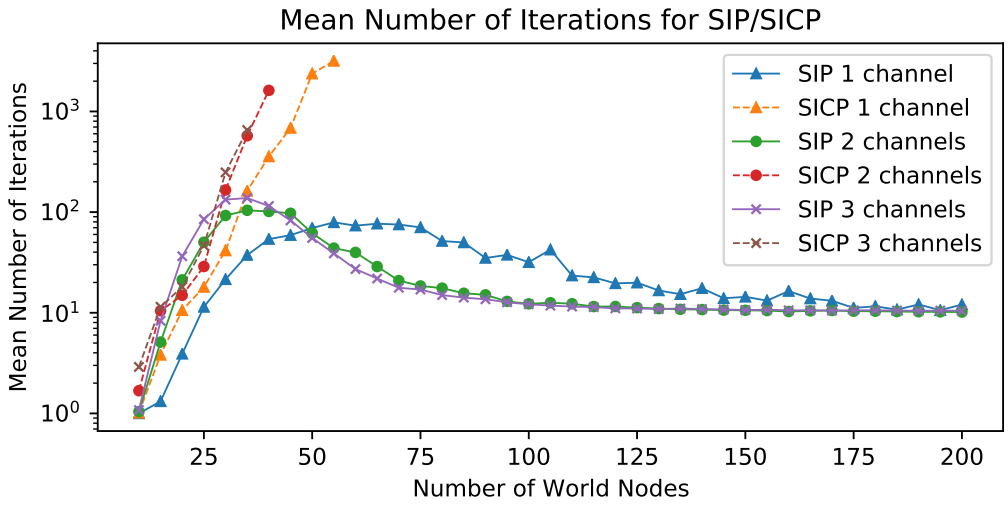


Figure 4.7: Mean number of search iterations (recursive calls to `IsoCount`) taken by Algorithm 13 to solve the SIP and SICP with one, two, and three channels as a function of world size. Results are averaged over 500 trials.

of world nodes. This is to be expected, as the expected number of SIs to be counted also scales monotonically with the number of world nodes, and the effort taken to count the SIs is closely related to the number of SIs.

4.4.3 Crosswords

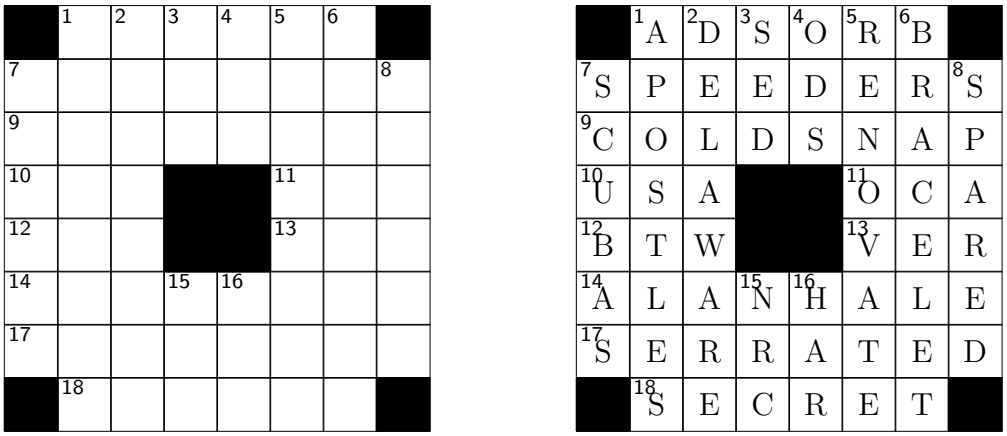
A crossword is a type of word puzzle typically consisting of a grid of black and white cells. For an example of a blank crossword grid and a solved crossword grid, see Figures 4.8a and 4.8b respectively. To solve a crossword, one must fill each cell with a letter so that each horizontal or vertical sequence of consecutive letters *answers* a corresponding *clue*. Each horizontal or vertical sequence of cells is identified by the number shown in its leftmost or topmost cell respectively and is read left-to-right or top-to-bottom respectively. For example, "10 Across" corresponds to the answer "USA" in the filled grid (Figure 4.8b), while "8 Down" corresponds to the answer "SPARED". Clues have been omitted for the crossword shown in Figure 4.8, but as an example, the clue for 10 Across could be "Country between Mexico and Canada, for short".

To design a crossword, a crossword constructor must fill a grid with answers and write a clue for each answer. Moreover, answers may only appear once in each puzzle and answers that intersect must share a letter at the point of intersection. For example, the answer for 1 Across must share its final letter with the first letter of the answer for 6 Down in Figure 4.8a. To aid in this process, specialized software is often used [Bur10]. Here, we observe that filling a crossword grid with answers can be viewed as a special case of the subgraph isomorphism problem and we use a specialized version of the Glasgow subgraph solver [MP15] to fill the crossword grid shown in Figure 4.8. We note that this is far from the first time filling a crossword grid has been viewed as a constraint satisfaction problem [RDM08, BCS01, Gin11], but it is the first time it has been viewed as a multiplex subgraph isomorphism problem to the best of our knowledge.

To view crossword construction as a special case of the subgraph isomorphism problem, we use the template to represent the crossword grid and the world to represent the dictionary of possible answers. We let each node in the template represent an entry in the grid that must be filled by an answer (e.g. 10 Across) and we connect nodes corresponding to intersecting entries by an edge whose channel indicates which letters must be shared between their answers. In the world, we let each node represent a possible answer (e.g. "USA") and we connect nodes with edges whose channels indicate which letters they share. With this definition of the template and world, a subgraph isomorphism yields a mapping from entries in the crossword grid to corresponding answers where each answer is used at most once and any intersecting answers share the necessary letter at their intersection. Thus, subgraph matching algorithms can be used to construct crosswords.

As a proof of concept, we convert the empty crossword grid shown in Figure 4.8a to a template network as described above. For the dictionary, we use the answers collected by Matthew L. Ginsberg [Gin11] restricted to those that occur at least 10 times. We do not convert the dictionary explicitly to a world network. Instead, we store the dictionary as a list of strings and compare characters between the strings to check compatibility with the

template edges. We adapted the Glasgow subgraph solver [MP15] to solve the multiplex subgraph isomorphism problem and to accept data in the format described. In about 30 seconds runtime, we are able to fill the crossword grid with the answers shown in Figure 4.8b.



(a) Empty crossword grid.

(b) Filled crossword grid.

Figure 4.8: An example of filling a crossword grid using a subgraph isomorphism algorithm.

4.4.4 Real-World Examples

We apply the algorithms to three real-world examples. From each dataset, we extract a small subgraph as our template and try to locate its matching subgraphs in the world graph. Table 4.7 summarizes the sizes and filtering results.

4.4.4.1 Great Britain Transportation

The Great Britain Transportation Network [GB15] combines the public transportation dataset available through the United Kingdom open-data program [Dep15] with timetables of domestic flights in the UK to obtain a multiplex time-dependent network that reflects the transportation network in the UK. There are six channels representing different transportation methods, including air, ferry, railway, metro, coach, and bus. This dataset has 262,377 nodes and 475,502 edges. The original dataset can be found at [GB15].

Dataset	Template		World		Channels	Number of isomorphisms	Filters	Problems solved
	Nodes	Edges	Nodes	Edges				
Britain Transportation	53	56	262,377	475,502	5	N/A	S, T, R, E	SIP
Britain Transportation (3 km)	53	56	262,377	475,502	5	3.76×10^7	L, S, T, R, E	SIP, SNSP, MCSP, SICP
Higgs Twitter	115	2,668	456,626	15,367,315	4	1.03×10^{14}	S, T, R	SIP, SNSP, MCSP, SICP
Commercial Airlines	37	210	450	7,177	37	3.65×10^9	S, T, R	SIP, SNSP, MCSP, SICP

Table 4.7: Sizes and filtering results on real-world examples in Sections 4.4.4.1, 4.4.4.2 and 4.4.4.3. The last column records the types of problems stated in Subsection 4.1.1 that we are able to solve. The names of the filters are abbreviated: L = Node Label; S = Node-level Statistics; T = Topology; R = Repeated-Sets; N = Neighborhood; E = Elimination.

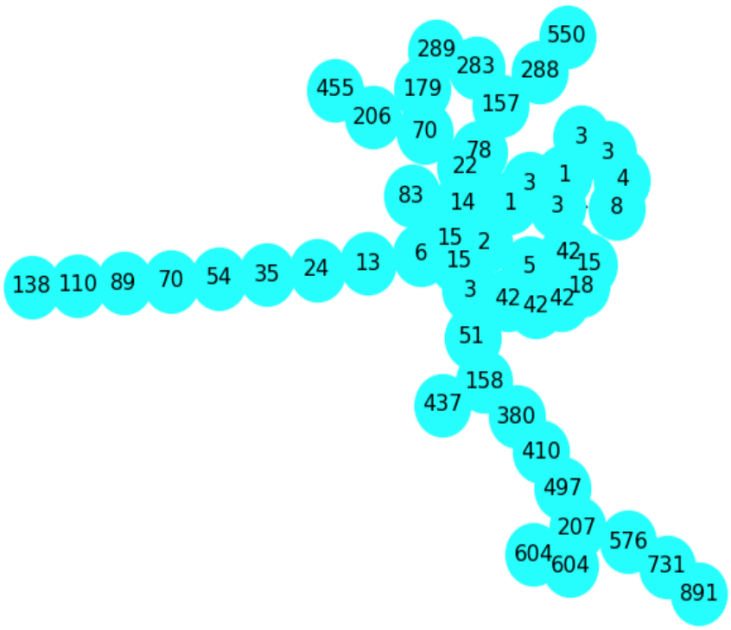


Figure 4.9: Candidate count for each node in the Great Britain Transportation template after applying the node-level statistics, topology, repeated-sets and elimination filters; without node label filtering.

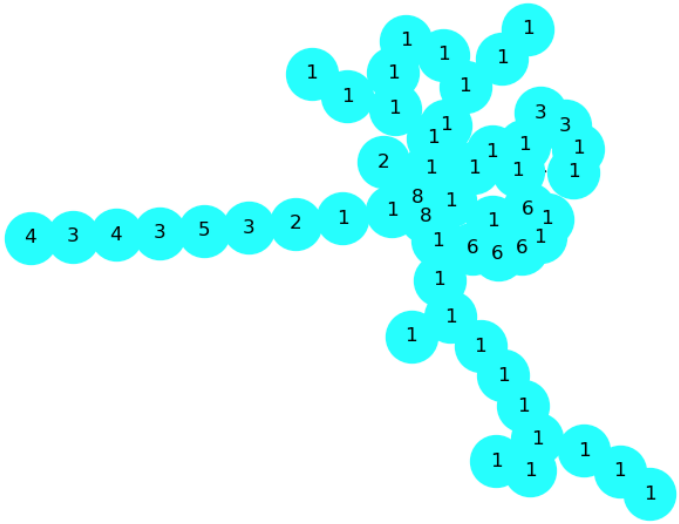


Figure 4.10: Candidate count for each node in the Great Britain Transportation template after applying the node label filter with a 3 km radius in addition to the node-level statistics, topology, repeated-sets, and elimination filters.

To test our algorithm, we first create a template. We identify a small set of locations that interact with each other through all channels (excluding airlines, since this channel is very sparse). If a location involves all five non-air channels in the network, we assume that it is important. There are only three nodes that interact in these 5 channels, and we chose one of them as our template center, which is Blackfriars Station in London. Starting from this node, we use a random walk to create a template, which has 53 nodes and 56 edges. The results are shown in Figure 4.9.

We can improve this result using the node label filter; the result is shown in Figure 4.10. In Figure 4.11, we see that the remaining candidates after filtering are geographically distributed across the UK. As previously mentioned in Subsection 4.2.1, these nodes are actually transportation stops with given latitude and longitudes. If we know *a priori* that a certain node should be confined in a region, say a radius of 3 km, then we can apply the node label filter. Node labels help reduce the size of the world before we apply filtering. Without the node label filter, even after the elimination filter, there are still too many candidates to

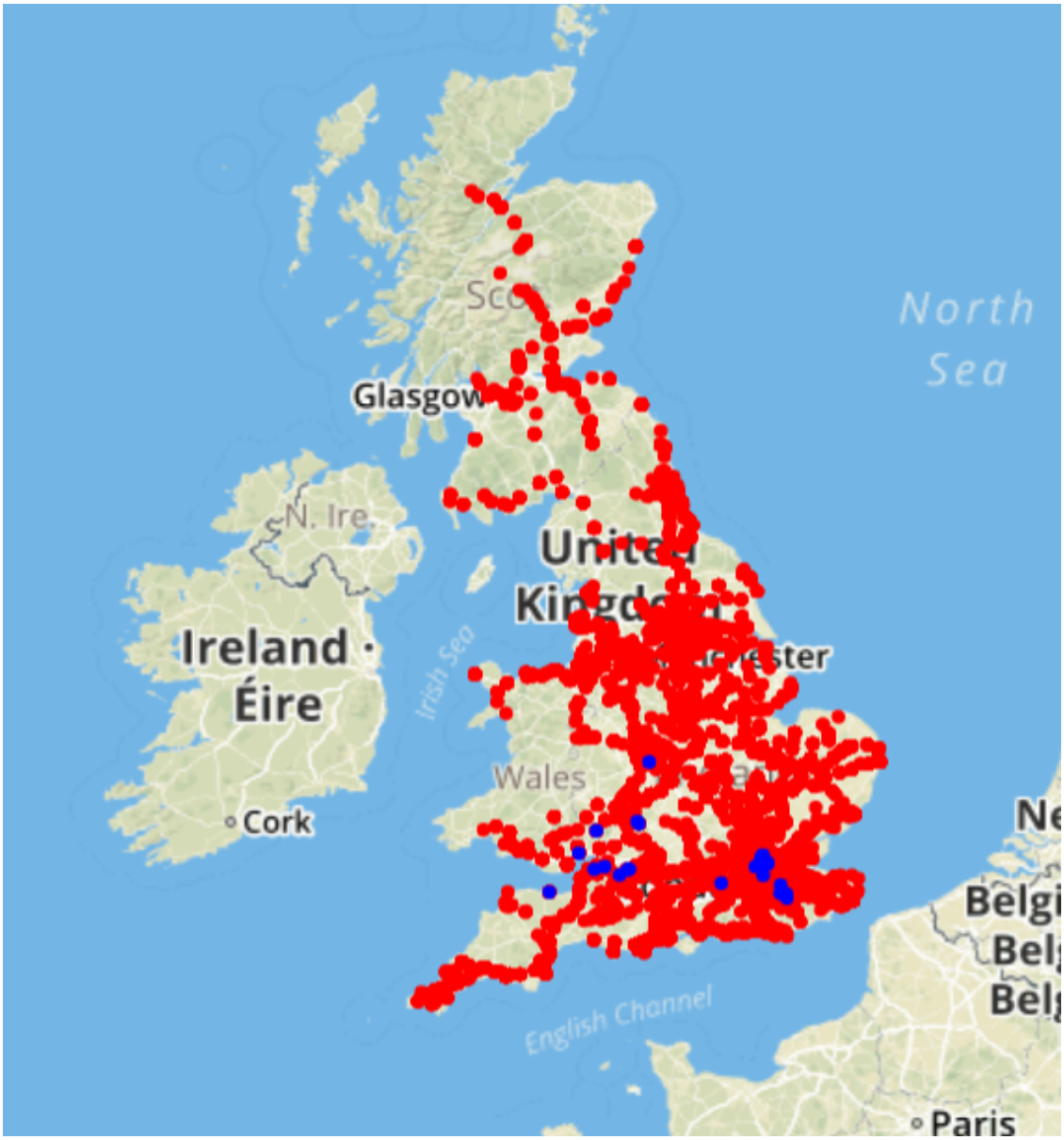


Figure 4.11: All candidates for the Great Britain Transportation dataset: blue nodes represent template nodes and red nodes represent candidates for the SIP.

count. However, if we apply geographical information before we run other filtering algorithms and restrict the candidates to be within 3 km from the coordinates of the template node, then we reduce the size of remaining candidates to a tractable level.

4.4.4.2 Higgs Twitter

The Higgs Twitter dataset [DLM13] records Twitter activities from July 1–7, 2012, during and after the discovery of the Higgs boson particle. It can be formulated into a directed, multiplex network with four channels, representing retweets, replies, mentions and follower relationships. The world graph contains 456,626 nodes (Twitter users) and 15,367,315 edges (interactions), and the user identities are aligned across all channels. To demonstrate how our filtering algorithm performs on detecting a small, relatively dense template, we select a group of 115 Twitter users frequently involved in retweets or replies during the week, and whose induced subgraph is connected, has multi-edges and contains edges in all four channels. The template has a total of 2,668 edges.

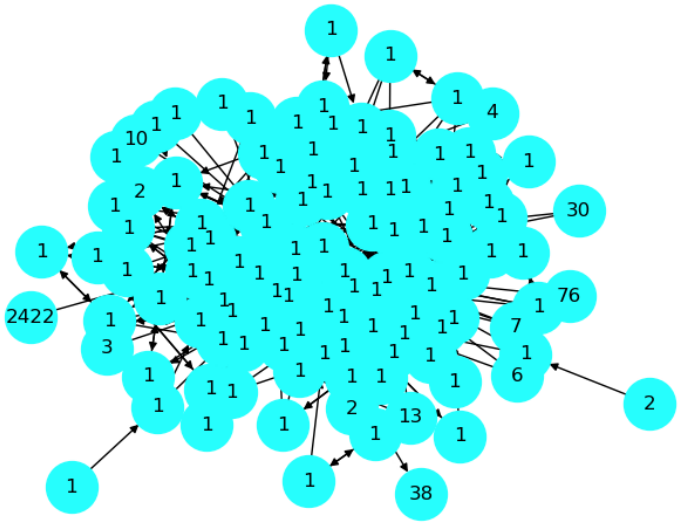


Figure 4.12: Candidate count for each node in the Higgs Twitter template after applying the node-level statistics, topology, and repeated-sets filters.

We apply our filtering methods to the world graph and the template. Our methods narrow down the candidates significantly and find exact matches for 102 out of the 115 template nodes (Figure 4.12). For the template nodes that have multiple remaining candidates, many of these candidates do belong to a valid subgraph isomorphism. We proceed to compute the number of subgraph isomorphisms, totaling 1.03×10^{14} valid isomorphisms. Our method works particularly well in detecting signals that exist across different channels and have multi-edges, as these graph properties provide us with enough information to remove invalid candidates.

4.4.4.3 Commercial Airlines

We also test our filtering methods on a commercial airlines dataset [CGZ13]. This dataset contains a multiplex airline network in Europe which consists of 37 channels, each representing a different airline. Compared with the other real world examples above, its world graph has a smaller scale, only containing 450 nodes (airports) and 7,177 directed, unweighted edges (flights).

We construct a template that has 37 nodes and 210 edges by taking the induced subgraph of 20 core nodes and introducing some additional periphery nodes and edges. When creating the template, we also ensure that it has nodes and edges across all channels to fully test the capability of our algorithm. Due to the small size of the dataset, our filtering algorithm is able to return the final candidate lists of all template nodes within a second, the result of which can be seen in Figure 4.13. The node-level statistics filter alone manages to find the exact match candidate of 10 templates, while the topology filter increases the number of exact matches to 28 nodes. Our counting algorithm also concludes that there exist around 3.6 million subgraph isomorphisms in the world graph. By using just the topological properties of the multiplex airline network, our algorithm is able to effectively identify the exact identities of most of the target airports as well as the potential identities of the remaining ones.

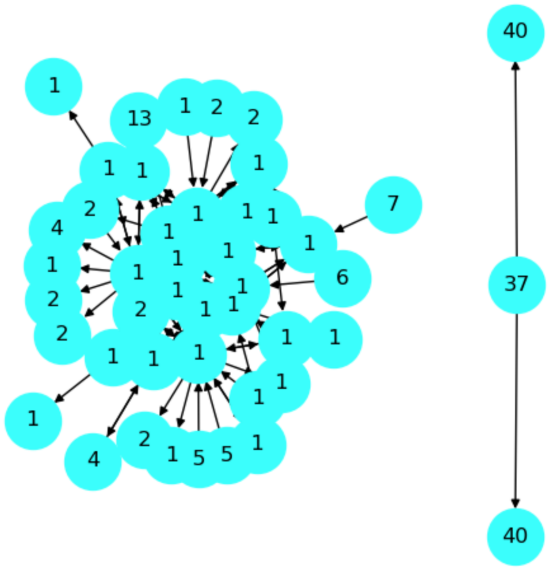


Figure 4.13: Candidate count for each node in the commercial airline template after applying the node-level statistics, topology and repeated-sets filters.

4.4.5 Adversarial Activity

We apply our filtering methods to a series of datasets developed for the DARPA MAA program [XTL18, DZJ18]. This program uses networks to represent activities (e.g., human trafficking, financial transactions, email communication, phone calls, distribution of narcotics) some of which may be adversarial. Because these activities can be covert, they may not be detectable through a single activity type. Multiplex networks provide a mathematical structure for identifying related network modalities for such complex adversarial actions. Thus, an important area of research is to match like patterns from an activity template to part of a larger dataset of multiplex actions. Here, we present some results on datasets created by three different teams: (1) Pacific Northwest National Laboratory (PNNL), (2) the Graphing Observables from Realistic Distributions In Activity Networks (GORDIAN) team, and (3) IvySys Technologies. These datasets consist of a 1,000–200,000 node world graph with one or more roughly 100-node templates, that simulate a scenario of activities by a specific group of agents.

Dataset	Instance	Template		World		Channels	Number of Isomorphisms	Filters	Problems Solved
		Nodes	Edges	Nodes	Edges				
	B0-S0	74	1,620	22,996	12,318,861	7	1,152	S, T, R	SIP, SNSP, MCSP, SICP, SMP
PNNL	B5-S0	64	1,201	22,994	12,324,975	7	1,152	S, T, R	SIP, SNSP, MCSP, SICP, SMP
Version 6	B1-S1	75	1,335	22,982	12,324,340	7	1,152	S, T, R, E	SIP, SNSP, MCSP, SICP, SMP
	B7-S1	81	1,373	23,011	12,327,168	7	3.13×10^8	S, T, R, E	SIP, SNSP, MCSP, SICP
PNNL Real World		35	158	6,407	74,862	3	2.12×10^{12}	S, T, R, N, E	SIP, SNSP, MCSP, SICP
GORDIAN	Batch-1	156	3,045	190,869	123,267,100	10	4.27×10^{15}	S, T, R, E	SIP, SNSP, MCSP, SICP
Version 7	Batch-2	44	715	190,869	123,264,754	10	1.35×10^{16}	S, T, R, E	SIP, SNSP, MCSP, SICP
IvySys Version 7		92	195	2,488	5,470,970	3	N/A	S, T, R, N, E	SIP
IvySys Version 11		103	387	1,404	5,719,030	5	N/A	S, T, R, E	SIP

Table 4.8: Overview of the sizes and filtering results of different DARPA datasets. For each instance of the datasets, the table records its basic statistics, which filters have been applied, and the number of isomorphisms. The last column states the types of problems stated in Subsection 4.1.1 that we can solve for each instance. The names of the filters are abbreviated: L = Node Label; S = Node-level Statistics; T = Topology; R = Repeated-Sets; N = Neighborhood; E = Elimination.

Table 4.8 summarizes the sizes and filtering results of each dataset. In each instance, there is one world graph with an embedded signal that is isomorphic to the template. We identify the signal with our filtering methods, and when there are multiple signals, we solve the SICP or MCSP to understand the solution space. For all of the datasets, we have applied the node-level statistics and topology filters to reduce the number of candidates. When the candidate counts remain high, we apply the neighborhood and elimination filters to further narrow them down.

4.4.5.1 PNNL Version 6

PNNL Version 6 [CPM18] has 12 instances, each consisting of one world and two templates. The worlds have around 23,000 nodes and over 12,000,000 edges each. The templates have 74–81 nodes and 1,200–1,650 edges.

By filtering based on node-level statistics, topology and repeated-sets, we identify the signals for most of the given templates. However, there remain templates (e.g., instances B1-S1, B7-S1 according to Table 4.8) where we cannot solve the MCSP by applying the previous filters. Under such circumstances, we additionally apply the elimination filter, which narrows down the candidate counts further as shown in Figure 4.14.

Figure 4.14 gives an overview of how different filters gradually reduce the number of candidates for each template node in instance B1-S1. In the bottom histogram, we observe that after node-level statistics and topology filters, around 5,000 world nodes still remain candidates of at least one template node. However, the subsequent application of elimination filter reduces the number of candidate world nodes to roughly 100. Figure 4.15 shows the comparison of filtering results on instance B1-S1 before and after adding in the elimination filter. The sharp contrast in their candidate counts shows that the elimination filter sometimes reduces the number of candidates significantly. After applying all filters, we observe in Figure 4.15 that some of the template nodes are permutable and share the same candidates, which lead to 1,152 SIs in this dataset.

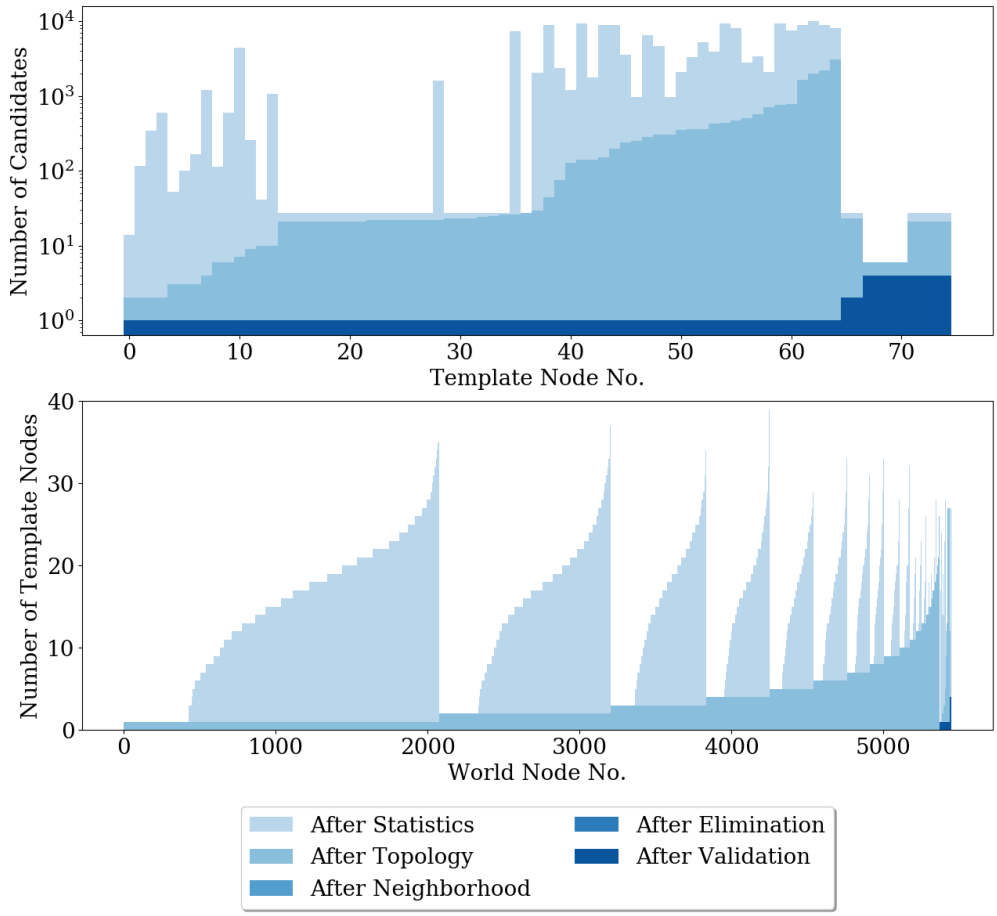
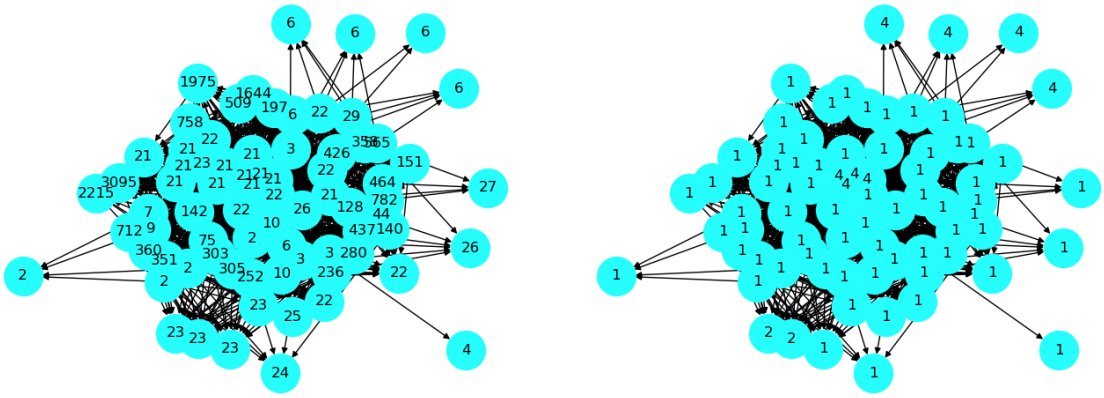


Figure 4.14: (Top): The number of candidates for each template node after different levels of filtering are applied to PNNL Version 6 B1-S1. (Bottom): The number of template nodes for which each world node is a candidate.. Note that the Validation histogram perfectly overlaps the Elimination histogram and the Neighborhood histogram perfectly overlaps the Topology histogram.



(a) Without elimination filter.

(b) With elimination filter.

Figure 4.15: Candidate count for each node in the PNNL Version 6 B1-S1 template after applying the node-level statistics, topology, and repeated-sets filters, with and without additionally applying the elimination filter.

The experiments on different instances of the PNNL Version 6 dataset demonstrate the efficacy of node-level statistics and topology filters for solving subgraph matching problems. The filtering results on the more difficult instances also display the potential of the elimination filter, especially in tackling problems that initially appear resistant to the node-level statistics and topology filters.

4.4.5.2 PNNL Real World

This dataset was created by PNNL from a social media dataset collected by Matteo Magnani and Luca Rossi [CDM10, MR11]. It involves friend/follower relationships on three social media platforms, each of which corresponds to a channel.

After applying all of the filters, we identify unique candidates for 20 template nodes. Many of the remaining template nodes have hundreds of candidates. By additionally applying validation to solve the MCSP, we verify that all of the remaining candidates participate in at least one signal. This suggests that the filters have the potential to reduce the candidates all the way to the solution of the MCSP.

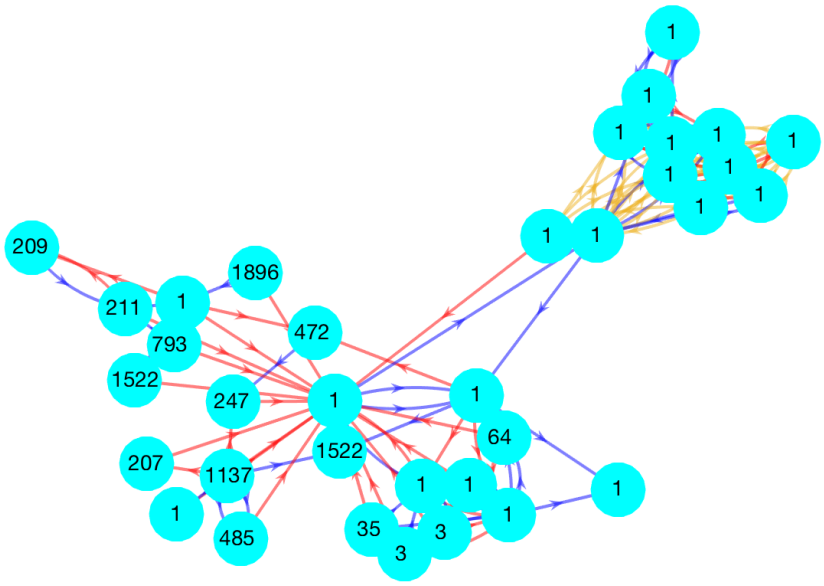


Figure 4.16: Candidate count for each node in the PNNL Real World template after solving the MCSP using validation. Edge colors correspond to their channels.

Both histograms in Figure 4.17 demonstrate how candidates get eliminated as different filters are applied. All filters are able to reduce the number of candidates. In the top histogram, the number of remaining candidates after the elimination filter entirely overlaps with that after validation, confirming that the result after the elimination filter is nearly identical to the solution to the MCSP, identifying all world nodes that correspond to each template node in at least one signal. In the bottom histogram, we can clearly observe how each filter eliminates the candidacy of world nodes in stages.

4.4.5.3 GORDIAN Version 7

GORDIAN Version 7 [KSG18] has two instances: Batch-1 and Batch-2. The Batch-1 template has 156 nodes and 3,045 edges, while the Batch-2 template has 44 nodes and 715 edges. Both instances have worlds with 190,869 nodes and about 123,265,000 edges. Despite the distinction in sizes and structures of their respective templates, both instances can be solved with node-level statistics, topology and repeated-sets filters. For example, in Batch-1, even

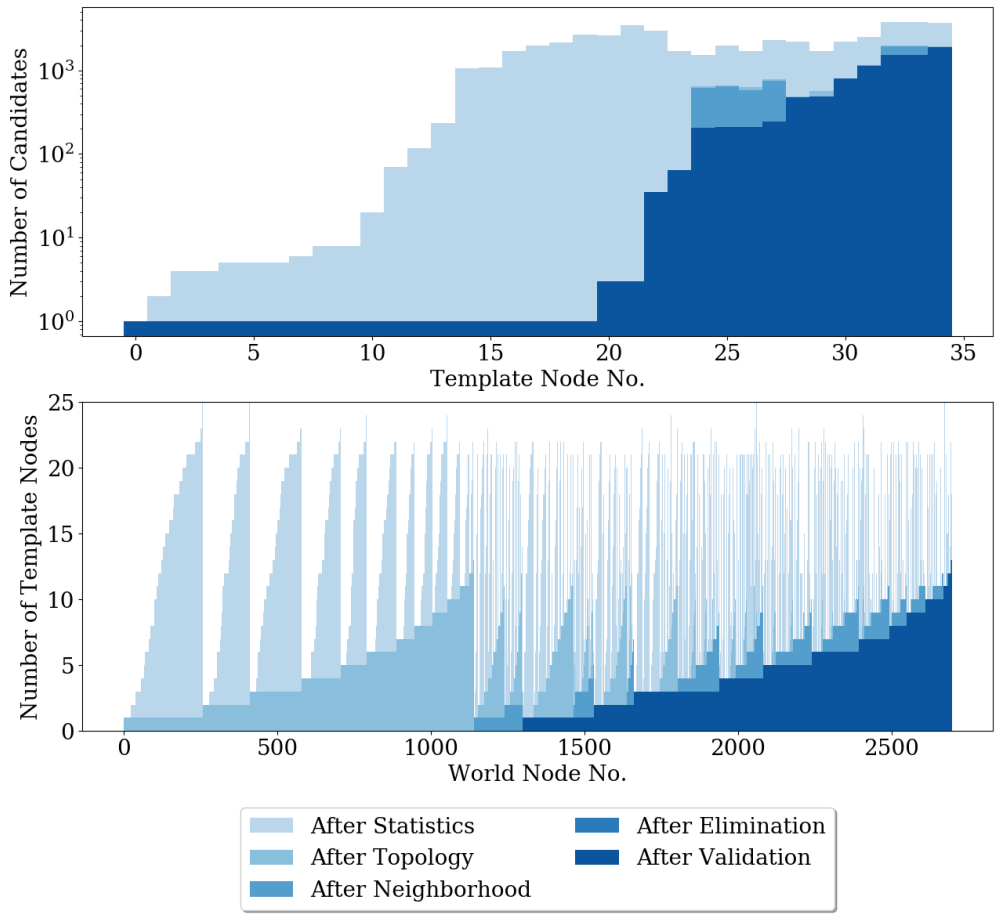


Figure 4.17: (Top): The number of candidates for each template node after different levels of filtering are applied to PNNL Real World. (Bottom): The number of template nodes for which each world node is a candidate. Note that the Validation histogram almost perfectly overlaps the Elimination histogram.

without applying the more advanced elimination and neighborhood filters, we manage to find the exact match for 129 template nodes out of 156. In Figure 4.18, we plot the largest connected component of instance Batch-1, in which we can clearly see that the majority of the template nodes are matched with their single candidate. These filtering results again demonstrate the potential of our basic filters (node-level statistics, topology, repeated-sets) in narrowing down the solution to subgraph matching problems. The small number of remaining candidates also enables us to apply validation and verify that all remaining candidates participate in some signal. We also compute the number of isomorphisms to be 1.51×10^{12}, and the enormous number here is a direct result of the permutability of some template nodes.

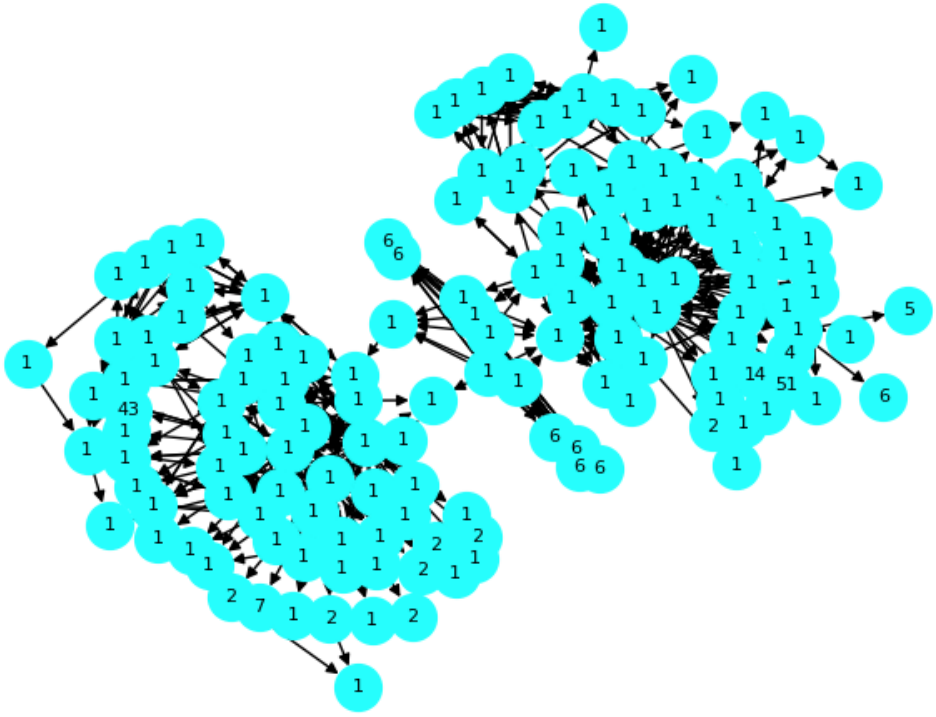


Figure 4.18: Candidate count for each node in the GORDIAN Version 7 Batch-1 template after applying the node-level statistics, topology, repeated-sets, and elimination filters.

4.4.5.4 IvySys Version 7

IvySys Version 7 [BJU18] has three channels corresponding to financial, communication and logistics transactions. Due to the sparsity of the template relative to that of the world, we are unable to identify any template nodes with unique candidates after applying different levels of filtering methods, as seen in Figure 4.19. However, after our filtering methods reduce the size of the search space, we find that there are in fact many signals in the world graph, some of which are almost completely disjoint from each other. It is their existence that leads to the abundance of candidates for each template node. Our approach allows us to easily solve the subgraph isomorphism problem (SIP) for IvySys Version 7, which is finding just one valid signal. However, the complexity of the solution space makes it unreasonable to proceed to solve the rest of the variants, like SICP, SNSP and MCSP.

4.4.5.5 IvySys Version 11

IvySys Version 11 consists of a single instance with five channels. It is somewhat similar to that of IvySys Version 7, as illustrated in Figure 4.20. The template has 103 nodes and 387 edges, while the world has 1,404 nodes and 5,719,030 edges. Even after the application of all of our existing filters, we can only identify exact matches for 13 out of the 103 template nodes. The majority of the template nodes still have a considerable number of candidates. A closer examination of Figure 4.20 shows the existence of permutable sets of template nodes. This suggests the possibility that we might have already reduced the graph to the point where almost all of the remaining candidates serve as part of a valid signal.

4.5 Conclusion

We have introduced a series of filtering methods that can be used to attack subgraph isomorphism problems within a large-scale multiplex network. The filtering approaches reduce the search space using node attributes, node-level statistics, topology, and alldifferent

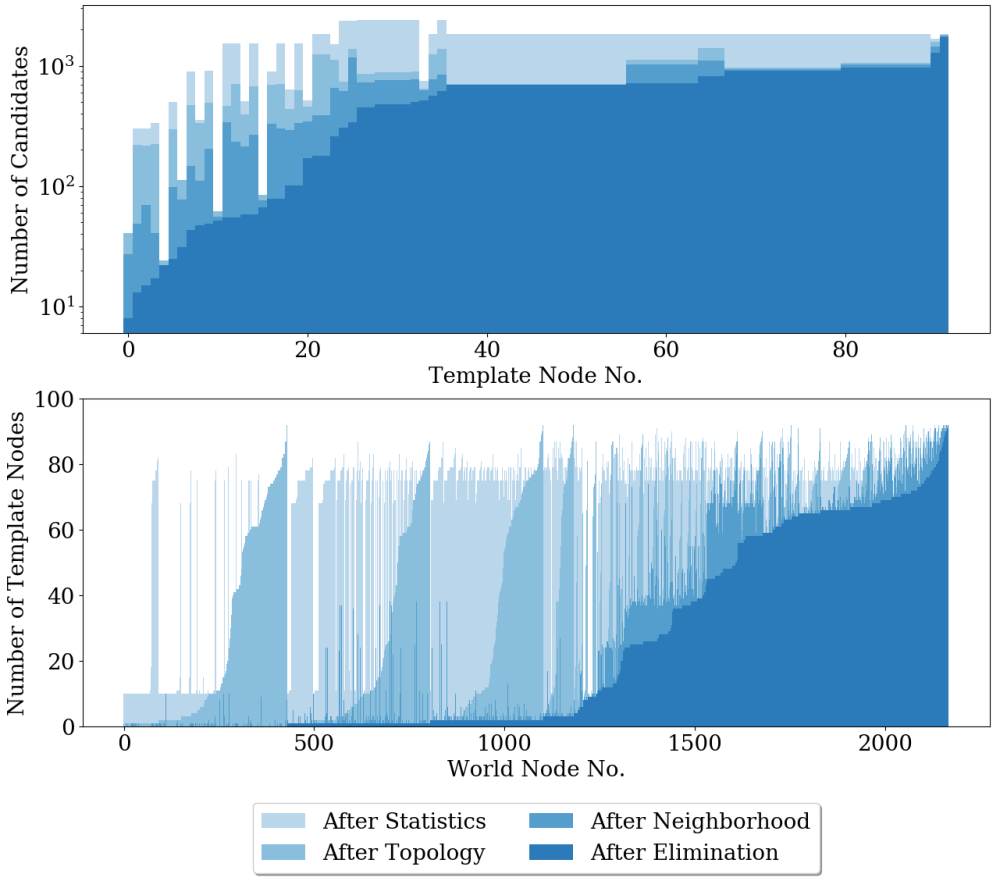


Figure 4.19: (Top): The number of candidates for each template node after different levels of filtering are applied to IvySys Version 7. (Bottom): The number of template nodes for which each world node is a candidate.

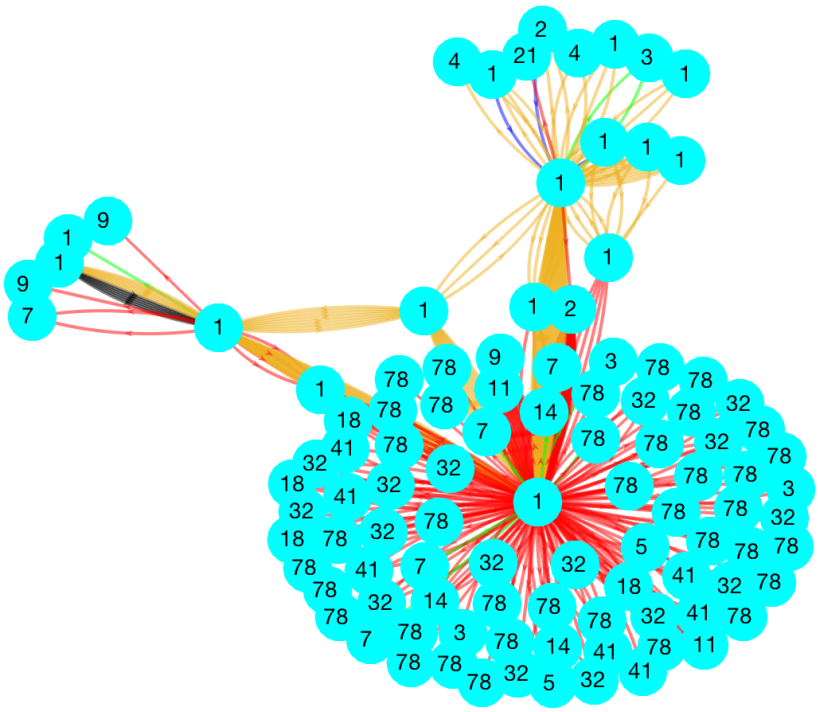


Figure 4.20: Candidate count for each node in the IvySys Version 11 template after applying the node-level statistics, topology, repeated-sets, and elimination filters. The colors of the edges correspond to their different channels.

constraints. When applied iteratively until convergence, our filtering methods can significantly narrow down the search space. We also implement a more computationally expensive isomorphism counting approach, which can be applied after filtering to solve the Subgraph Isomorphism Counting Problem (Definition 4.1.5). We can also proceed with a validation step that checks whether each world node participates in at least one subgraph isomorphism, hence solving the Signal Node Set Problem (Definition 4.1.6).

We have applied our methods to multiple types of networks: Sudoku puzzles, three real-world datasets, and synthetic datasets created by PNNL, GORDIAN and IvySys as part of the DARPA MAA program. In all of these experiments, our methods have greatly narrowed down the candidates of the template nodes and provided us with an understanding of the size of the solution space. When applied to the Higgs Twitter network, the commercial airline network, PNNL Version 6, PNNL Real World and GORDIAN Version 7, our methods are capable of solving most of the proposed subgraph isomorphism problems. However, even in settings where we cannot solve the problems, such as the Great Britain Transportation and IvySys Versions 7 and 11, our methods reduce the search space to a much smaller scale. Our results show that further narrowing down the candidates of template nodes is often hindered by an abundance of subgraph isomorphisms existing in the world graphs. Under such circumstances, leveraging more node attributes such as geospatial coordinates or timestamps can assist in making further progress, as shown with the Great Britain Transportation example.

CHAPTER 5

Inexact Attributed Subgraph Matching*

5.1 Introduction

Subgraph matching involves searching for a prescribed *template* graph as a subgraph of a *world* graph. One may want to find one occurrence, all occurrences, or some other summary of the solution space, depending on the application [CFS03]. In applications where the graphs have attributes on the nodes and edges, it is common to search for subgraphs of the world which only approximately match the template. This is called *inexact attributed subgraph matching*.

Definition 5.1.1 (Attributed Graph)**.** *An* attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$ *consists of a set of nodes* $\mathcal{V}$*, a set of edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$*, a set of attributes* $\mathcal{A}$*, and a map* $\mathcal{L} : \mathcal{V} \cup \mathcal{E} \to \mathcal{A}$ *from nodes and edges to their attributes.*

Given two attributed graphs, a template $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{L}_t, \mathcal{A})$ and a world $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{L}_w, \mathcal{A})$, we are interested in one-to-one maps $f : \mathcal{V}_t \to \mathcal{V}_w$ where the induced subgraph of the image is similar to the template. Such a map f is called an *inexact match* of $\mathcal{G}_t$. The cost metric used to measure similarity between the template and the image of f is denoted $C(f; \mathcal{G}_t, \mathcal{G}_w)$ and is described in Subsection 5.2.1.

The process of finding the map $f^\star$ which minimizes $C(f; \mathcal{G}_t, \mathcal{G}_w)$ is called inexact *attributed subgraph isomorphism (ASI)*, while the process of finding all maps f with distance at most ϵ

*This chapter is adapted from [TMY20].

is called inexact *attributed subgraph matching (ASM)*. Finding the k most similar maps is called *top-k inexact ASM*.

In the remainder of this section, we discuss existing approaches to ASI/ASM and inexact ASI/ASM, as well as our contributions. In Section 5.2, we describe our approach to solving inexact ASI, inexact ASM, and top-k inexact ASM. In Section 5.3, we evaluate our methods on the AIDA V2.1.2 dataset which consists of knowledge graphs collected from news articles about political events in the Ukraine.

5.1.1 Related Work

Algorithms for exact subgraph matching are discussed in Subsection 4.1.2. Algorithms for inexact subgraph matching are more diverse than those for exact subgraph matching. Different applications and research areas define the inexact subgraph matching problem in different ways. Some algorithms take tree search, constraint propagation, or graph indexing approaches similar to that seen in exact subgraph matching [PTT14, JHW19, KX19, LZ17], while other algorithms relax the discrete optimization problem into a continuous one in order to apply traditional continuous optimization techniques such as gradient descent [SPP20].

5.1.2 Contributions

We introduce algorithms for inexact attributed subgraph isomorphism and matching to find optimal subgraphs as measured by graph edit distance. We show results for noisy queries on knowledge graphs collected from news articles about political events in the Ukraine.

5.2 Algorithm

In Subsection 5.2.1, we define the cost metric $C(f; \mathcal{G}_t, \mathcal{G}_w)$ that will be minimized in inexact ASI/ASM. In Subsections 5.2.2 and 5.2.3, we explain how to compute lower bounds on $C(f; \mathcal{G}_t, \mathcal{G}_w)$ under the constraint $f(t_c) = w_c$ for each $t_c \in \mathcal{V}_t$ and $w_c \in \mathcal{V}_w$. Using these

constrained lower bounds, in Subsection 5.2.4 we describe a tree search procedure for inexact ASI/ASM.

5.2.1 Graph Edit Distance Based Cost Metric

The graph edit distance [SF83, GXT10] is a measure of the distance between two graphs, as defined by the total cost of the cheapest sequence of edits which transform one graph into another. The six possible edits usually considered are node addition, node deletion, node substitution, edge addition, edge deletion, and edge substitution. Here, we consider a cost metric using only the latter four edits.

The node substitution costs are measured by a user-defined function $D_{\mathcal{V}} : (\mathcal{V}_t \times \mathcal{V}_w) \to \mathbb{R}^+$ that typically compares the labels of the nodes. A common node substitution function is

$$D_{\mathcal{V}}(v_t, v_w) = \mathbb{1}\left[\mathcal{L}_t(v_t) \neq \mathcal{L}_w(v_w)\right]$$

so that $\sum_{t \in \mathcal{V}_t} D_{\mathcal{V}}(t, f(t))$ counts the number of node assignments which do not preserve the node label. If the node labels belong to a normed vector space, the corresponding norm can be used

$$D_{\mathcal{V}}(v_t, v_w) = \|\mathcal{L}_t(v_t) - \mathcal{L}_w(v_w)\|.$$

Likewise, the edge substitution, addition and deletion costs are measured by user-defined functions $D : (\mathcal{E}_t \times \mathcal{E}_w) \to \mathbb{R}^+$, $D^+ : \mathcal{E}_w \to \mathbb{R}^+$ and $D^- : \mathcal{E}_t \to \mathbb{R}^+$. The edge related cost functions are combined into a single function $D_{\mathcal{E}}$ for brevity

$$D_{\mathcal{E}}(e, f(e)) = \begin{cases} D(e, f(e)), & e \in \mathcal{E}_t, f(e) \in \mathcal{E}_w, \\ D^-(e), & e \in \mathcal{E}_t, f(e) \notin \mathcal{E}_w, \\ D^+(f(e)), & e \notin \mathcal{E}_t, f(e) \in \mathcal{E}_w, \\ 0, & e \notin \mathcal{E}_t, f(e) \notin \mathcal{E}_w. \end{cases} \tag{5.1}$$

For convenience, we use the shorthand $f((t_1, t_2)) = (f(t_1), f(t_2))$ so that $f(e)$ is well-defined. Given a template $\mathcal{G}_t$, a world $\mathcal{G}_w$, and a map $f : \mathcal{V}_t \to \mathcal{V}_w$, the cost metric $C(f; \mathcal{G}_t, \mathcal{G}_w)$ is

defined as

$$C(f;\mathcal{G}_t,\mathcal{G}_w)=\sum_{t\in\mathcal{V}_t}D_{\mathcal{V}}(t,f(t))+\sum_{e\in\mathcal{V}_t\times\mathcal{V}_t}D_{\mathcal{E}}(e,f(e)). \tag{5.2}$$

5.2.2 Cost Bounds

We now discuss lower bounds on the cost metric. The cost metric $C(f;\mathcal{G}_t,\mathcal{G}_w)$ can be decomposed as

$$C(f;\mathcal{G}_t,\mathcal{G}_w)=\sum_{t\in\mathcal{V}_t}L(t,f(t);\mathcal{G}_t,\mathcal{G}_w,f),$$

where $L(t,f(t);\mathcal{G}_t,\mathcal{G}_w,f)$ are local costs related to each template node t and matching world node $f(t)$. We use decompositions where the local cost L has the form

$$L(t,w;\mathcal{G}_t,\mathcal{G}_w,f)=D_{\mathcal{V}}(t,w)+\frac{1}{2}\sum_{t_o\in\mathcal{V}_t}\Big[D_{\mathcal{E}}\big((t,t_o),(w,f(t_o))\big)+D_{\mathcal{E}}\big((t_o,t),(f(t_o),w)\big)\Big]. \tag{5.3}$$

This way, the local cost $L(t,w;\mathcal{G}_t,\mathcal{G}_w,f)$ captures the cost incurred by assigning world node w to template node t and half of the cost of assigning the edges incident to w to those incident to t. Edge costs are halved so that the local costs sum to the correct total cost $C(f;\mathcal{G}_t,\mathcal{G}_w)$; otherwise, edge costs would be counted twice, once for each endpoint. If the edge addition cost were neglected (i.e. $D^+\equiv 0$), the summation $\sum_{t_o\in\mathcal{V}_t}$ reduces to $\sum_{t_o\in\mathcal{N}_t}$, where $\mathcal{N}_t$ is the set of neighbors of t. One often useful variation is to alter these factors independently for each edge to place more importance on certain nodes; we omit this discussion for now, but refer to Section 5.A for the details.

To bound the full cost, we start by bounding the local cost $L(t,w;\mathcal{G}_t,\mathcal{G}_w,f)$ from below by $B(t,w;\mathcal{G}_t,\mathcal{G}_w)$ defined as follows

$$\begin{aligned}B(t,w;\mathcal{G}_t,\mathcal{G}_w)&:=D_{\mathcal{V}}(t,w)+\frac{1}{2}\sum_{t_o\in\mathcal{V}_t}\min_{w_o\in\mathcal{V}_w}\Big(D_{\mathcal{E}}\big((t,t_o),(w,w_o)\big)+D_{\mathcal{E}}\big((t_o,t),(w_o,w)\big)\Big)\\&\le L(t,w;\mathcal{G}_t,\mathcal{G}_w,f).\end{aligned} \tag{5.4}$$

Note that the local bound $B(t,w;\mathcal{G}_t,\mathcal{G}_w)$ has no dependence on the map f. Within the context of inexact ASI/ASM, the set $\mathcal{V}_w$ iterated over in the minimization can often be

reduced for practical purposes; we refer to Section 5.B for the details. We can extend the bound on the local cost to a naive bound on the full cost

$$\begin{aligned} G_{\text{Naive}}(\mathcal{G}_t, \mathcal{G}_w) &:= \sum_{t \in \mathcal{V}_t} \min_{w \in \mathcal{V}_w} B(t, w; \mathcal{G}_t, \mathcal{G}_w) \\ &\leq \sum_{t \in \mathcal{V}_t} B(t, f(t); \mathcal{G}_t, \mathcal{G}_w) \\ &\overset{\text{Eqn. (5.3)}}{\leq} \sum_{t \in \mathcal{V}_t} L(t, f(t); \mathcal{G}_t, \mathcal{G}_w, f) = C(f; \mathcal{G}_t, \mathcal{G}_w). \end{aligned} \tag{5.5}$$

Alternatively, we can compute a tighter lower bound

$$\begin{aligned} G_{\text{LAP}}(\mathcal{G}_t, \mathcal{G}_w) &:= \min_{\substack{g: \mathcal{V}_t \to \mathcal{V}_w \\ \text{s.t. } g \text{ is 1-1}}} \sum_{t \in \mathcal{V}_t} B(t, g(t); \mathcal{G}_t, \mathcal{G}_w) \\ &\leq \sum_{t \in \mathcal{V}_t} B(t, f(t); \mathcal{G}_t, \mathcal{G}_w) \\ &\overset{\text{Eqn. (5.3)}}{\leq} \sum_{t \in \mathcal{V}_t} L(t, f(t); \mathcal{G}_t, \mathcal{G}_w, f) = C(f; \mathcal{G}_t, \mathcal{G}_w). \end{aligned} \tag{5.6}$$

by leveraging the fact that f must be one-to-one (1-1). We refer to G_{Naive} and G_{LAP} as "global cost bounds".

Computing the global cost bound G_{LAP} is equivalent to solving a rectangular linear assignment problem (LAP) of size $|\mathcal{V}_t| \times |\mathcal{V}_w|$. Many algorithms exist to solve the LAP and its rectangular variant. The most notable algorithms include the Hungarian algorithm [Kuh55], the Munkres algorithm [Mun57], the Jonker–Volgenant (JV) algorithm [JV87], and the auction algorithm [Ber88]. In our implementation, we apply a modified JV algorithm [Cro16], which is designed to efficiently solve rectangular LAPs. The time complexity of the solver is $O(|\mathcal{V}_t|\,|\mathcal{V}_w|^2)$. In contrast, the time required to compute the naive global cost bound G_{Naive} is only $O(|\mathcal{V}_t|\,|\mathcal{V}_w|)$.

5.2.3 Constrained Cost Bounds

To use the global cost bounds from Equations (5.5) and (5.6) in our algorithm, we must compute the bounds under the constraint $f(t_c) = w_c$ for each $t_c \in \mathcal{V}_t$ and each $w_c \in \mathcal{V}_w$.

These constrained global cost bounds will be used as a heuristic for performing a tree search described in Subsection 5.2.4. The corresponding constrained global cost bounds are

$$G^c_{\text{Naive}}(t_c, w_c; \mathcal{G}_t, \mathcal{G}_w) := B(t_c, w_c; \mathcal{G}_t, \mathcal{G}_w) + \sum_{\substack{t \in \mathcal{V}_t \\ t \neq t_c}} \min_{\substack{w \in \mathcal{V}_w \\ w \neq w_c}} B(t, w; \mathcal{G}_t, \mathcal{G}_w) \tag{5.7}$$

for the naive global cost bound and

$$G^c_{\text{LAP}}(t_c, w_c; \mathcal{G}_t, \mathcal{G}_w) := \min_{\substack{g: \mathcal{V}_t \to \mathcal{V}_w \\ \text{s.t. } g \text{ is 1-1} \\ \text{s.t. } g(t_c) = w_c}} \sum_{\substack{t \in \mathcal{V}_t \\ t \neq t_c}} B(t, f(t); \mathcal{G}_t, \mathcal{G}_w) \tag{5.8}$$

for the LAP-based global cost bound.

To compute these constrained global cost bounds, the first step in either case is to compute the local bounds $B(t, w; \mathcal{G}_t, \mathcal{G}_w)$ for each $t \in \mathcal{V}_t$ and $w \in \mathcal{V}_w$. This takes $O\left(|\mathcal{V}_t|\,|\mathcal{V}_w|\,\text{avgtime}(B)\right)$ time, where avgtime(B) denotes the mean time to compute the local bound for a fixed pair of nodes $t, w \in \mathcal{V}_t \times \mathcal{V}_w$. The remaining calculations to compute Equation (5.7) for each $t_c \in \mathcal{V}_t$ and $w_c \in \mathcal{V}_w$ can be done in $O(|\mathcal{V}_t|\,|\mathcal{V}_w|)$ time. Computing Equation (5.8) is more complicated.

To compute Equation (5.8) we must solve a LAP of size $(|\mathcal{V}_t|-1)\times(|\mathcal{V}_w|-1)$ for each $t_c \in \mathcal{V}_t$ and $w_c \in \mathcal{V}_w$. Solving each of these LAPs separately would require $|\mathcal{V}_t|\,|\mathcal{V}_w|$ applications of the $O(|\mathcal{V}_t|\,|\mathcal{V}_w|^2)$ LAP solver for a total of $O(|\mathcal{V}_t|^2\,|\mathcal{V}_w|^3)$ compute time. However, this approach can be improved since each LAP is a constrained version of the same unconstrained LAP in Equation (5.6).

To start, we solve the unconstrained LAP in Equation (5.6) to find the map

$$g_\star = \operatorname*{arg\,min}_{\substack{g: \mathcal{V}_t \to \mathcal{V}_w \\ \text{s.t. } g \text{ is 1-1}}} \sum_{t \in \mathcal{V}_t} B(t, g(t); \mathcal{G}_t, \mathcal{G}_w)$$

in $O(|\mathcal{V}_t|\,|\mathcal{V}_w|^2)$ time. Next, we use the dynamic Hungarian algorithm [KSD07] to enforce the constraint $g(t_c) = w_c$ on the unconstrained map $g_\star$ for each $t_c \in \mathcal{V}_t$ and $w_c \in \mathcal{V}_w$. Enforcing the constraint $g(t_c) = w_c$ frees up $g_\star(t_c)$ to potentially be reassigned to some other template node. When w_c is not in the image of $g_\star$, reassigning $g_\star(t_c)$ costs $O(|\mathcal{V}_t|^2)$ for each $t_c \in \mathcal{V}_t$,

independent of w_c, for a total of $O(|\mathcal{V}_t|^3)$. When w_c is in the image of $g_\star$, the constraint $g(t_c) = w_c$ also frees up $g_\star^{-1}(w_c)$ to be reassigned to some other world node.

5.2.4 Search for Optimal Solutions

From Subsection 5.2.3, we have a procedure for computing lower bounds on $C(f; \mathcal{G}_t, \mathcal{G}_w)$ under the constraint $f(t_c) = w_c$ for each $t_c \in \mathcal{V}_t$ and $w_c \in \mathcal{V}_w$. Now, we treat these lower bounds as a heuristic for performing a greedy depth first search. For each template node t, we assign $f(t) = w$ for the candidate w with the lowest bound, then recompute the bounds under that additional assignment. We assign candidates to template nodes with the fewest minimum bound candidates first, as this gives an indication of which template nodes have only a few "good" choices. After assigning all template nodes in this way, we obtain a map f.

Although f may not be the optimal map $f^\star$ that we seek, it can be used to drastically cut down on the list of possible assignments we have to consider going forward. We compute the cost of f to serve as an upper bound on the cost of the optimal map $f^\star$. We keep track of the *cost threshold* U and set it equal to the smallest cost $C_{\min}$ we have seen so far, with $f_{\min}$ the corresponding map. Using $C_{\min}$, the search space is refined by skipping assignments $f(t) = w$ which lead to lower bounds that are greater than or equal to $C_{\min}$, since in inexact ASI we are only interested in the map $f^\star$ which minimizes the cost.

After identifying a new map or eliminating all remaining possibilities due to the cost bounds, we backtrack and try assigning different world nodes to some template nodes. This is done until there are no options left to explore, at which point we have found the optimal map $f^\star = f_{\min}$.

To perform inexact ASM instead of inexact ASI, one can fix $U \equiv \epsilon$ and record all of the maps f that are observed during the search. In this context, we only skip assignments when their cost bound is strictly greater than ϵ, so that we do not accidentally skip matches whose cost is exactly ϵ.

To perform top-k inexact ASM, keep track of $f_1, \ldots, f_k$ and $C_1, \ldots, C_k$ which track the

k best maps seen so far and their costs. Use C_k instead of $C_{\min}$ to prune the search space, and only prune when the cost bound is greater than or equal to C_k. When the search is completed, $f_1, \ldots, f_k$ are the k maps with the lowest cost.

One way to speed up the search in inexact ASI and top-k inexact ASM is to set an initial value for U. However, this approach can lead to no solutions being found if the chosen value is lower than the cost of the optimal solution $C(f^\star; \mathcal{G}_t, \mathcal{G}_w)$.

5.3 Experiments

We test our algorithms on the AIDA Version 2.1.2 dataset created by Pacific Northwest National Laboratory (PNNL) for the DARPA–MAA program. This dataset consists of a knowledge graph collected from news articles about political events in the Ukraine. Provided also is a measure of distance between attributes, which can be used to construct the corresponding graph edit distance. The world graph has 98,817 nodes and 138,127 edges. Three templates are provided, each with six different variations labeled A-F. These templates are much smaller than the world graph, consisting of 11-33 nodes and 11-40 edges. These templates were created by taking a known "ground truth" existing subgraph and adding increasing levels of noise. The A template has no noise and is guaranteed to have at least one exact match. Provided also is the ground truth mapping for the A version of each template. All experiments were run on an HP Z8 G4 Workstation with two 12-core 6,136 3.0 2,666MHz CPUs.

We perform top-k inexact ASM on these templates, with $k = 5$. We use two approaches: the first approach (labeled "Normal") in the table, simply runs the algorithm with no initial cost threshold, while the second uses the ground truth cost bound (labeled "GTCB") from the A template to derive an appropriate initial cost threshold for the other templates in each series (1,2,3). To identify this threshold, we first impose the matching from the ground truth, then set our algorithm to find optimal assignments for any remaining nodes that were not included in the ground truth. Using only this initial cost threshold, we discard all

Table 5.1: Results for the AIDA Version 2.1.2 dataset, showing time taken and the cost of the best match that was found. The algorithm was cut off if it failed to complete within 46.5 hours, taking the best match that it had found so far.

	Time		Cost		
Template	Normal	GTCB	Normal	Ground Truth	GTCB
1A	27.8 min	1.5 sec	0.0	0.0	0.0
1B	24.5 min	0.2 sec	0.347	0.351	0.347
1C	46.5 hrs	5.48 hrs	8.783	3.610	2.254
1D	46.5 hrs	64.1 min	15.470	1.639	1.636
1E	46.5 hrs	35.9 hrs	16.194	2.642	2.638
1F	46.5 hrs	76.7 min	13.596	2.328	1.772
2A	2.25 min	0.06 sec	0.0	0.0	0.0
2B	3.07 min	0.15 sec	0.307	0.335	0.307
2C	6.35 min	10.6 sec	4.070	4.891	4.070
2D	12.4 min	32.9 sec	3.839	5.942	3.839
2E	4.58 min	22.4 sec	3.848	4.533	3.848
2F	18.5 min	42.3 sec	4.012	4.704	4.012
3A	2.33 min	0.08 sec	0.0	0.0	0.0
3B	2.27 sec	0.15 sec	0.145	0.191	0.145
3C	4.10 min	3.56 sec	2.452	2.452	2.452
3D	10.3 min	2.48 min	2.312	2.585	2.312
3E	20.8 min	10.9 min	2.472	3.471	2.472
3F	4.80 min	0.94 sec	1.314	1.348	1.314

other matching information from the ground truth and proceed to optimize over the space of matches with lower cost than the cost of the ground truth. The cost of the ground truth for each template is listed in the Ground Truth column under Cost. In real world contexts, the ground truth is unknown. However, it is not unreasonable that another, possibly suboptimal, approximate match exists that we can use for the purpose of setting the initial cost threshold to restricting the solution space.

The results of the two approaches on the AIDA Version 2.1.2 dataset are shown in Table 5.1. The algorithm was cut off if it failed to complete within 46.5 hours, taking the best match that it had found so far. Although we perform top-k matching, we list only one cost for each template; this is because for all templates considered, all 5 matches found were of the same cost. We observe that for all of the A templates, both the normal and GTCB approaches converge to the expected result of an exact match with 0 graph edit distance. For all templates other than 1C-F, both approaches were able to complete within the limit of 46.5 hours, and arrive at the optimal solutions. For 1C-F, the normal version hit the runtime limit, yielding the best solutions found within that time. For the GTCB version, only template 1E was aborted early, after 35.9 hours. This was done because, when examining the branching structure of the search algorithm, the approach was deemed unlikely to complete. The other three templates completed and reached optimal solutions.

After identifying optimal solutions (for all templates except 1E), we then apply the approach for inexact ASM discussed in Subsection 5.2.4 to find all optimal solutions by setting the cost threshold to the cost of an optimal solution. The number of optimal solutions found is listed in Table 5.2. In the case of 1E, we set this cost threshold to be the cost of the best solution found; we observe that all found solutions were of the same cost.

For templates 1D, 1E, and 1F, the optimal solution search was cut off after 209 hours due to time and memory limitations. We believe that there exist more solutions than those that were found, possibly orders of magnitude more. We have marked these templates with a $>$ in Table 5.2 to indicate this.

5.3.1 Analysis of Solution Space

For real world applications it is important to understand the solution space beyond simply finding one match. For example, in security applications, the match may point to specific people or places, some of which could be imposters. We show an example here of a map from the template to the full solution space — something that could be visualized for analysts trying to understand the connectivity of the template nodes in the full solution space. After using the methods discussed in prior sections to discover subgraph matchings of minimal cost, we can appeal to symmetries apparent in the template graph and world graph to attain a more compact representation of the solution space. Symmetry in graph structures has been studied in depth in the context of subgraph matching [HLL13, RW15, BCL16, NYG19]. It is a confounding factor for subgraph discovery that can lead to redundant work while exploring symmetric areas of the graph. The kind of symmetry most often utilized is where nodes are considered *structurally equivalent* if they have the same label and are connected to the same neighbors by edges with the same attributes [Sai79]. For an example of this type of symmetry, see Figure 5.1.

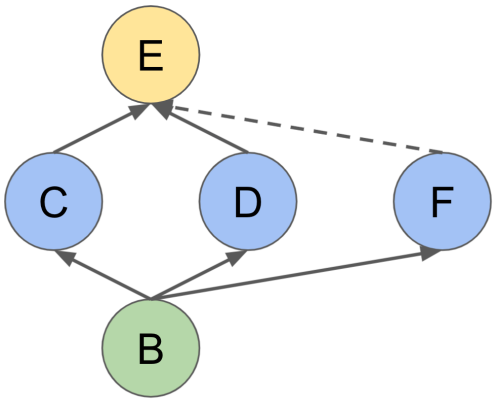


Figure 5.1: Structural Equivalence: Nodes C and D are structurally equivalent and F is equivalent to neither. C, D, and F have the same node label and same set of neighbors. However, the edge connecting F to E has a different label than the edges connecting C and D to E, so F is not equivalent to C or D.

In the matching problems on the AIDA dataset, symmetry in the graph accounts for the combinatorial explosion in solutions for certain template graphs, especially in instances where more noise is present. Figure 5.2 demonstrates various forms of symmetry in the world graph and how they affect the structure of the solution space for Template 1B. We present the template graph and its matches to a subgraph of the world graph containing only those nodes appearing in an optimal solution. We color each node in the template graph the same as its candidates in the world graph. The blue and orange nodes in the world graph are two groups of structurally equivalent nodes and so can be swapped out arbitrarily in any solution and maintain a matching of the same cost. The red and lavender groups of world nodes present a slightly more complex form of symmetry (automorphic). Exploiting it for purposes of generating more solutions would require a more intelligent scheme that assigns the red and lavender template nodes as a unit. We leave automorphic symmetry for future work and only consider structural equivalence in this chapter. In this example, the problem has 6,120 optimal cost solutions; applying structural equivalence, we can reduce it to 2,520 solutions from which we can generate the rest.

An a posteriori analysis of the solutions generated by our search enables us to compress the solution space as well as to illustrate potential ways to significantly speed up the subgraph search. We take the subgraph of world nodes that appear in a minimal cost solution and compute groups of nodes that are structurally equivalent. Then to compress the solution space, we eliminate any solution that can be generated by swapping out equivalent world nodes in another mapping. We do this by examining the classes of solutions generated by interchanging structurally equivalent nodes and identifying a representative from each.

Table 5.2 demonstrates the extent to which structural equivalence reduces the description of the solution space to a smaller set. We also list the number of world nodes that appear in a minimal cost solution as well as the number of structural equivalence classes to show the level of equivalence in the solution space. If the number of equivalence classes is significantly smaller than the number of world nodes, there is a great deal of equivalence. We observe

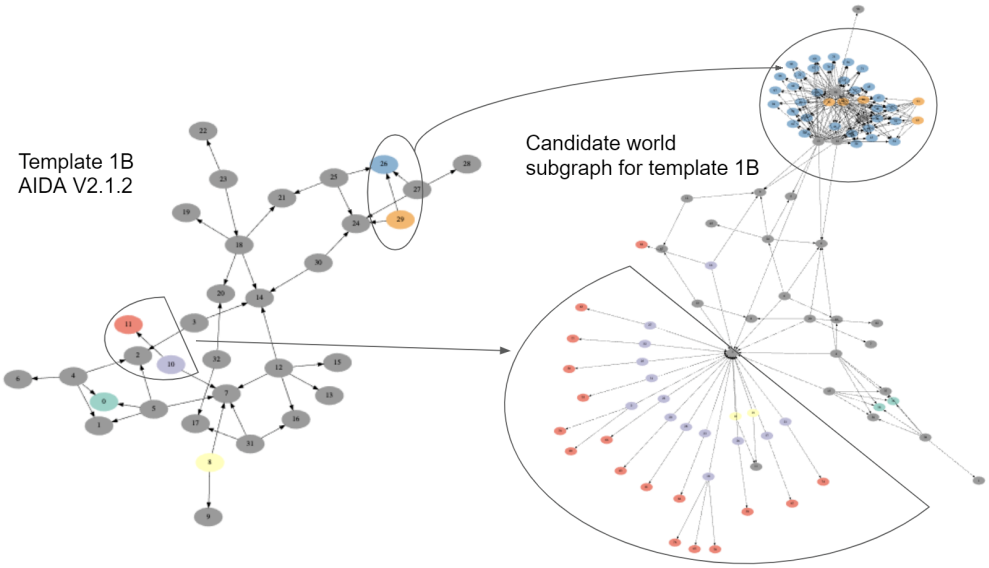


Figure 5.2: Template graph and world subgraph for Problem 1B. Colored groups of nodes in the world graph are candidates for nodes of the same color in the template graph. The long arrows and shapes denote corresponding groupings.

that in certain cases, e.g., templates 1E and 1F, the size of the representative solution set is nearly ten times smaller. If an algorithm were to efficiently compute these symmetries and incorporate them in a subgraph search, then we might expect speedups of an order of magnitude for these problems.

An analysis through a symmetry lens also exposes how introducing noise into a subgraph matching problem impacts the solution space. Broadly, if more information is known about the labels and neighbors of vertices in both the template and world, there will be less symmetry apparent in the matching problem. This can be seen in Table 5.2 with increased noise as we go from A to F which significantly expands the solution space. Intuitively, having more label information allows nodes to distinguish themselves from each other and break symmetry. If we introduce noise into a problem, say by removing a template node's label, then the labels of the world graph become irrelevant as the label cost will be the same.

The graphs provided in the AIDA datasets have three different labels: "rdf:type", which indicates the semantic type of a node (e.g. Person, Location, Vehicle, etc.), "hasName", which

Table 5.2: The number of solutions, representative solutions, candidate world nodes, and equivalence classes for each subgraph matching problem from the AIDA Version 2.1.2 dataset. For templates 1D-F, the code was terminated due to runtime constraints before all solutions could be found.

	Template					
	1A	1B	1C	1D	1E	1F
# Solutions	6	6,120	324	>392k	>382k	>400k
# Rep. Sols.	3	2,520	162	>315k	>34k	>58k
# World Nodes	38	99	47	109	3,169	3,141
# Eq. Classes	37	95	46	105	1,686	1,692
	2A	2B	2C	2D	2E	2F
# Solutions	78	1,400	39	13,780	1,248	17,368
# Rep. Sols.	6	60	3	1,160	128	4,160
# World Nodes	29	41	27	45	37	48
# Eq. Classes	17	23	15	30	26	36
	3A	3B	3C	3D	3E	
# Solutions	6	6	36	198	198	
# Rep. Sols.	4	4	16	92	92	
# World Nodes	16	16	21	26	23	
# Eq. Classes	15	15	18	23	23	

gives the names of entities, and "textValue", which contains miscellaneous text information associated with the node. Table 5.3 lists the number of equivalence classes for the first set of templates when considering only the "rdf:type" label as compared to considering all labels. As can be seen, when considering only the type label, we have significantly fewer equivalence

classes. This is especially apparent in template 1E and 1F for which the minimal cost solution requires two edge mismatches leaving an isolated template node that may match any world graph node as long as the label matches. Of course, when constructing a matching, we must consider all label information; however, understanding the relationship between different levels of noise and symmetry appears critical to fully understanding the solution space of a given problem.

Table 5.3: Equivalence classes when considering only part of the label and the full label

Template	1A	1B	1C	1D	1E	1F
# Type Eq. Classes	37	67	45	73	36	67
# Full Eq. Classes	37	95	46	105	1,686	1,692

5.4 Conclusion

We provided an approach for inexact subgraph matching on attributed graphs via optimization of the graph edit distance. Using an approach analogous to constraint propagation, we developed lower bounds on the cost of individual assignments, then combined them using linear sum assignment. Using these cost bounds as a heuristic, we performed a guided depth first search for optimal solutions. We applied our approach to the AIDA V2.1.2 knowledge graph dataset.

In the future, we hope to extend our method to more general types of graph templates, such as templates with pairwise relative attribute constraints. For example, one could impose a constraint that requires two nodes have attribute values whose difference lies within a minimum and maximum range. This is important in the context of temporal data, where two events may be required to occur within a certain time window of each other. Similarly, for spatial data, it may be useful to require two nodes to be within a certain distance of each other.

5.A Varying Edgewise Weights

In some situations, the costs of certain nodes become more important than other nodes. For example, when a node has a known assignment, it is less important to know the cost associated with that node, and more important to know the costs of its surrounding nodes, so that they may too be assigned candidates using the best possible heuristic.

We repeat the cost bound formulation from Equation (5.3), but replace $\frac{1}{2}$ by a parameter function $\alpha(t, t_o)$

$$L_\alpha(t, w; \mathcal{G}_t, \mathcal{G}_w, f) = D_\mathcal{V}(t, w) + \sum_{t_o \in \mathcal{V}_t} \alpha(t, t_o) \Big(D_\mathcal{E}\big((t, t_o), (w, f(t_o))\big) + D_\mathcal{E}\big((t_o, t), (f(t_o), w)\big) \Big).$$

The new parameter function $\alpha(t, t_o)$ has the property that $\alpha(t, t_o) + \alpha(t_o, t) = 1$ under the constraint $0 \leq \alpha(t, t_o) \leq 1 \forall t, t_o$. If we wish to assign t more importance than t_o, we use $\alpha(t, t_o) = 1, \alpha(t_o, t) = 0$.

Similarly, we extend $B(t, w; \mathcal{G}_t, \mathcal{G}_w)$ to $B_\alpha(t, w; \mathcal{G}_t, \mathcal{G}_w)$ defined as follows

$$B_\alpha(t, w; \mathcal{G}_t, \mathcal{G}_w) := D_\mathcal{V}(t, w) + \sum_{t_o \in \mathcal{V}_t} \alpha(t, t_o) \min_{w_o \in \mathcal{V}_w} \Big(D_\mathcal{E}\big((t, t_o), (w, w_o)\big) + D_\mathcal{E}\big((t_o, t), (w_o, w)\big) \Big).$$

In practice, the modified local cost bound above is used during the search algorithm to put less weight on assigned nodes. When a node is given an assignment, it is also given a relative weight of $\alpha = 0$, while its neighbors are given a weight of $\alpha = 1$ with respect to that node. For an edge between two assigned nodes, we default to $\alpha = \frac{1}{2}$.

5.B Restricting Candidates During Minimization

Within the context of the search approach detailed in Subsection 5.2.4, at certain points in the algorithm, we have an upper bound U on the cost. At this point in the algorithm, the exact value of the cost bound is not needed if it is greater than or equal to U (or strictly greater in the context of inexact ASM).

Thus, we can instead define the local cost bound as

$$B(t, w; \mathcal{G}_t, \mathcal{G}_w) :=$$
$$\min\Big(U, D_{\mathcal{V}}(t, w) + \sum_{t_o \in \mathcal{V}_t} \alpha(t, t_o) \min_{w_o \in C(t_o)} \Big[D_{\mathcal{E}}\big((t, t_o), (w, w_o)\big) + D_{\mathcal{E}}\big((t_o, t), (w_o, w)\big)\Big]\Big)$$

where $C(t_o)$ is defined to be the set of world nodes w_o for which the known constrained cost bound $G(t_o, w; \mathcal{G}_t, \mathcal{G}_w)$ is strictly less than U (less than or equal to for inexact ASM). Doing so drastically improves computational performance and provides a tighter bound on costs which lie below U. This also refines cost bounds analogously to constraint propagation; as tighter global cost bounds G are found, this leads to tighter local cost bounds, which are then used to compute even tighter global cost bounds until we reach a final set of bounds.

CHAPTER 6

Conclusion

In this thesis, we studied iterative methods for solving linear systems of equations and specialized algorithms for pattern matching on multiplex networks. These algorithms are of fundamental importance as subroutines in numerical linear algebra, statistics, data analysis, cheminformatics, knowledge bases, and other areas. Our analysis and experiments cast light on how to choose the appropriate algorithm for a given problem and how to specialize it to desired applications.

We analyzed the convergence of sketch-and-project methods with adaptive sampling strategies. According to our new convergence theory, the greediest strategies enjoy the fastest convergence guarantees among all such strategies. We also analyzed the convergence of averaged Kaczmarz methods with fixed sampling strategies. As the number of samples in the averaging increases, we see that the rate of convergence improves and the convergence horizon for inconsistent systems decreases. We demonstrated empirically that the convergence of the adaptive sketch-and-project methods and averaged Kaczmarz methods reflects our worst-case convergence theory.

We discussed subgraph matching and various related problems related to searching graph data. We introduced filtering algorithms that are specialized to multiplex networks. Finding one match is typically much easier than finding all matches, but for some patterns we were still able to find or count all subgraphs matching the pattern. When the problems were relaxed to allow subgraphs that only *approximately* match the pattern, we modified the filtering approach to identify subgraphs which match as closely as possible. In both the exact and approximate settings, we observed that there are often a combinatorially large number

of matching subgraphs depending on the amount of symmetry of the pattern and data.

Solving linear systems of equations and graph search algorithms are important subroutines in numerous areas. Improvements in these subroutines translate directly to improvements in corresponding applications. Looking forward, one avenue for further improvements would be to design and analyze distributed methods. Such methods would be particularly useful when data is not centralized to a single location. Another avenue would be carefully implementing the methods discussed as high-performance computing programs. These advances would further broaden the types and increase the scale of data to which the methods could be applied in practice.

[AC89] R. Aharoni and Y. Censor. "Block-iterative projection methods for parallel computation of solutions to convex feasibility problems." *Linear Algebra and its Applications*, **120**:165–175, 1989.

[ADH08] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. C. Sahinalp. "Biomolecular network motif counting and discovery by color coding." *Bioinformatics*, **24**(13):i241–i249, 2008.

[AG18] B. K. Abid and R. Gower. "Stochastic algorithms for entropy-regularized optimal transport problems." In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1505–1512. PLMR, 2018.

[ALS15] G. Alain, A. Lamb, C. Sankar, A. Courville, and Y. Bengio. "Variance reduction in SGD by distributed importance sampling." *arXiv preprint arXiv:1511.06481*, 2015.

[ANR15] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. "Efficient graphlet counting for large networks." In *2015 IEEE International Conference on Data Mining*, pp. 1–10. IEEE, 2015.

[BBB16] B. Bentley, R. Branicky, C. L. Barnes, Y. Chew, E. Yemini, E. T. Bullmore, P. E. Vértes, and W. R. Schafer. "The multilayer connectome of caenorhabditis elegans." *PLoS Computational Biology*, **12**(12):e1005283, 2016.

[BCL16] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang. "Efficient subgraph matching by postponing Cartesian products." In *Proceedings of the 2016 International Conference on Management of Data*, pp. 1199–1214. ACM, 2016.

[BCN18] L. Bottou, F. E. Curtis, and J. Nocedal. "Optimization methods for large-scale machine learning." *Siam Review*, **60**(2):223–311, 2018.

[BCS01] A. Beacham, X. Chen, J. Sillito, and P. van Beek. "Constraint programming lessons learned from crossword puzzles." In *Advances in Artificial Intelligence*, pp. 78–87. Springer, 2001.

[BE85] R. Bar-Yehuda and S. Even. "A local-ratio theorem for approximating the weighted vertex cover problem." In *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pp. 27–45. Elsevier, 1985.

[Ber88] D. P. Bertsekas. "The auction algorithm: A distributed relaxation method for the assignment problem." *Annals of Operations Research*, **14**:105–123, 1988.

[BGP13] V. Bonnici, R. Giugno, A. Pulvirenti, D. Shasha, and A. Ferro. "A subgraph isomorphism algorithm and its application to biochemical data." *BMC Bioinformatics*, **14**(7):S13, 2013.

[BJU18] K. O. Babalola, O. B. Jennings, E. Urdiales, and J. A. DeBardelaben. "Statistical methods for generating synthetic email data sets." In *2018 IEEE International Conference on Big Data*, pp. 3986–3990. IEEE, 2018.

[BN15] J. Briskman and D. Needell. "Block Kaczmarz method with inequalities." *Journal of Mathematical Imaging and Vision*, **52**(3):385–396, 2015.

[Bot98] L. Bottou. "Online algorithms and stochastic approximations." In *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[Bur10] J. Burston. "Crossword compiler ver. 8.1." *CALICO Journal*, **28**(1):175–190, 2010.

[BW18a] Z.-Z. Bai and W.-T. Wu. "On greedy randomized Kaczmarz method for solving large sparse linear systems." *SIAM Journal on Scientific Computing*, **40**(1):A592–A606, 2018a.

[BW18b] Z.-Z. Bai and W.-T. Wu. "On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems." *Applied Mathematics Letters*, **83**:21–26, 2018b.

[BW19a] Z.-Z. Bai and W.-T. Wu. "On greedy randomized coordinate descent methods for solving large linear least-squares problems." *Numerical Linear Algebra with Applications*, **26**(4):e2237, 2019a.

[BW19b] Z.-Z. Bai and W.-T. Wu. "On partially randomized extended Kaczmarz method for solving large sparse overdetermined inconsistent linear systems." *Linear Algebra and its Applications*, **578**:225–250, 2019b.

[Byr07] C. L. Byrne. *Applied Iterative Methods*. Ak Peters/CRC Press, 2007.

[CDM10] F. Celli, F. M. L. Di Lascio, M. Magnani, B. Pacelli, and L. Rossi. "Social network data and practices: the case of FriendFeed." In *Advances in Social Computing*, pp. 346–353. Springer, 2010.

[CFS03] D. Conte, P. Foggia, C. Sansone, and M. Vento. "Graph matching applications in pattern recognition and image processing." In *International Conference on Image Processing*, volume 2, pp. II–24. IEEE, 2003.

[CFS04] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. "A (sub)graph isomorphism algorithm for matching large graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(10):1367–1372, 2004.

[CFS17] V. Carletti, P. Foggia, A. Saggese, and M. Vento. "Introducing VF3: A new algorithm for subgraph isomorphism." In *Graph-Based Representations in Pattern Recognition*, pp. 128–139. Springer, 2017.

[CFS18] V. Carletti, P. Foggia, A. Saggese, and M. Vento. "Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **40**(4):804–818, 2018.

[CFV15] V. Carletti, P. Foggia, and M. Vento. "VF2 plus: An improved version of VF2 for biological graphs." In *Graph-Based Representations in Pattern Recognition*, pp. 168–177. Springer, 2015.

[CGZ13] A. Cardillo, J. Gómez-Gardeñes, M. Zanin, M. Romance, D. Papo, F. del Pozo, and S. Boccaletti. "Emergence of network features from multiplexity." *Scientific Reports*, **3**(1344), 2013.

[CP12] X. Chen and A. M. Powell. "Almost sure convergence of the Kaczmarz algorithm with random measurements." *Journal of Fourier Analysis and Applications*, **18**(6):1195–1214, 2012.

[CPM18] J. A. Cottam, S. Purohit, P. Mackey, and G. Chin. "Multi-channel large network simulation including adversarial activity." In *2018 IEEE International Conference on Big Data*, pp. 3947–3950. IEEE, 2018.

[CQR15] D. Csiba, Z. Qu, and P. Richtarik. "Stochastic dual coordinate ascent with adaptive probabilities." In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 674–683. PLMR, 2015.

[CR18] D. Csiba and P. Richtárik. "Importance sampling for minibatches." *The Journal of Machine Learning Research*, **19**(1):962–982, 2018.

[Cro16] D. F. Crouse. "On implementing 2D rectangular assignment algorithms." *IEEE Transactions on Aerospace and Electronic Systems*, **52**(4):1679–1696, 2016.

[Cut13] M. Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport." In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pp. 2292–2300. Curran Associates, 2013.

[CZT12] Y. Cai, Y. Zhao, and Y. Tang. "Exponential convergence of a randomized Kaczmarz algorithm with relaxation." In *Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science*, pp. 467–473. Springer, 2012.

[Dep15] Department for Transport. "National public transport data repository." `https://data.gov.uk/dataset/d1f9e79f-d9db-44d0-b7b1-41c216fe5df6/`

national-public-transport-data-repository-nptdr, 2015. Accessed: 2021-05-30.

[Dev86] L. Devroye. *Non-Uniform Random Variate Generation*. Springer, 1986.

[DGL89] I. S. Duff, R. G. Grimes, and J. G. Lewis. "Sparse matrix test problems." *ACM Transactions on Mathematical Software*, **15**(1):1–14, 1989.

[DGL92] I. S. Duff, R. G. Grimes, and J. G. Lewis. "Users' guide for the Harwell–Boeing sparse matrix collection." Technical Report TR/PA/92/86, CERFACS, 1992.

[DH11] T. A. Davis and Y. Hu. "The University of Florida Sparse Matrix Collection." *ACM Transactions on Mathematical Software*, **38**(1):1–25, 2011.

[DLM13] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi. "The anatomy of a scientific rumor." *Scientific Reports*, **3**(2980), 2013.

[Dum15] B. Dumitrescu. "On the relation between the randomized extended Kaczmarz algorithm and coordinate descent." *BIT Numerical Mathematics*, **55**(4):1005–1015, 2015.

[DZJ18] J. Douglas, B. Zimmerman, A. K. ana J. Xu, D. Sussman, and V. Lyzinski. "Metrics for evaluating network alignment." In *11th ACM International Conference on Web Search and Data Mining, Workshop on Graph Techniques for Adversarial Activity Analytics*. ACM, 2018.

[EHL81] P. P. B. Eggermont, G. T. Herman, and A. Lent. "Iterative algorithms for large partitioned linear systems, with applications to image reconstruction." *Linear algebra and its Applications*, **40**:37–67, 1981.

[Elf80] T. Elfving. "Block-iterative methods for consistent and inconsistent linear equations." *Numerische Mathematik*, **35**:1–12, 1980.

[EN11] Y. C. Eldar and D. Needell. "Acceleration of randomized Kaczmarz method via the Johnson–Lindenstrauss lemma." *Numerical Algorithms*, **58**(2):163–177, 2011.

[Fan12] W. Fan. "Graph pattern matching revised for social network analysis." In *Proceedings of the 15th International Conference on Database Theory*, pp. 8–21. ACM, 2012.

[GB15] R. Gallotti and M. Barthelemy. "The multilayer temporal network of public transport in Great Britain." *Scientific Data*, **2**(140056), 2015.

[GBH70] R. Gordon, R. Bender, and G. T. Herman. "Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography." *Journal of Theoretical Biology*, **29**(3):471–481, 1970.

[GG05] D. Gordon and R. Gordon. "Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems." *SIAM Journal on Scientific Computing*, **27**(3):1092–1117, 2005.

[Gin11] M. L. Ginsberg. "Dr. Fill: Crosswords and an implemented solver for singly weighted CSPs." *Journal of Artificial Intelligence Research*, **42**:851–886, 2011.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co, 1979.

[GMM19] R. M. Gower, D. Molitor, J. D. Moorman, and D. Needell. "Adaptive sketch-and-project methods for solving linear systems." *arXiv preprint arXiv:1909.03604*, 2019.

[GMN08] I. P. Gent, I. Miguel, and P. Nightingale. "Generalised arc consistency for the AllDifferent constraint: An empirical survey." *Artificial Intelligence*, **172**(18):1973–2000, 2008.

[GO12] M. Griebel and P. Oswald. "Greedy and randomized versions of the multiplicative Schwarz method." *Linear Algebra and its Applications*, **437**(7):1596–1610, 2012.

[GR15a] R. M. Gower and P. Richtárik. "Stochastic dual ascent for solving linear systems." *arXiv preprint arXiv:1512.06890*, 2015a.

[GR15b] R. M. Gower and P. Richtárik. "Randomized iterative methods for linear systems." *SIAM Journal on Matrix Analysis and Applications*, **36**(4):1660–1690, 2015b.

[GR16] R. Gower and P. Richtárik. "Linearly convergent randomized iterative methods for computing the pseudoinverse." *arXiv preprint arXiv:1612.06255*, 2016.

[GR17] R. M. Gower and P. Richtárik. "Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms." *SIAM Journal on Matrix Analysis and Applications*, **38**(4):1380–1409, 2017.

[GV13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU press, 4 edition, 2013.

[GXT10] X. Gao, B. Xiao, D. Tao, and X. Li. "A survey of graph edit distance." *Pattern Analysis and applications*, **13**:113–129, 2010.

[HK73] J. E. Hopcroft and R. M. Karp. "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs." *SIAM Journal on Computing*, **2**(4):225–231, 1973.

[HLL13] W. Han, J. Lee, and J. Lee. "Turbo$_{\text{iso}}$: towards ultrafast and robust subgraph isomorphism search in large graph databases." In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 337–348. ACM, 2013.

[HM93] G. T. Herman and L. B. Meyer. "Algebraic reconstruction techniques can be made computationally efficient (positron emission tomography application)." *IEEE Transactions on Medical Imaging*, **12**(3):600–609, 1993.

[HN90a] M. Hanke and W. Niethammer. "On the acceleration of Kaczmarz's method for inconsistent linear systems." *Linear Algebra and its Applications*, **130**:83–98, 1990a.

[HN90b] M. Hanke and W. Niethammer. "On the use of small relaxation parameters in Kaczmarz method." *Zeitschrift fur Angewandte Mathematik und Mechanik*, **70**(6):T575–T576, 1990b.

[HN19] J. Haddock and D. Needell. "On Motzkin's method for inconsistent linear systems." *BIT Numerical Mathematics*, **59**(2):387–401, 2019.

[Hoe] W.-J. van Hoeve. "The AllDifferent constraint: A survey." http://www.andrew.cmu.edu/user/vanhoeve/papers/alldiff.pdf. Accessed: 2021-05-30.

[HS52] M. R. Hestenes and E. Stiefel. "Methods of conjugate gradients for solving linear systems." *Journal of Research of the National Bureau of Standards*, **49**(6):409–436, 1952.

[HS78] C. Hamaker and D. Solmon. "The angles between the null spaces of x rays." *Journal of Mathematical Analysis and Applications*, **62**(1):1–23, 1978.

[HS08] H. He and A. Singh. "Graphs-at-a-time: query language and access methods for graph databases." In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of data*, pp. 405–418. ACM, 2008.

[HWP03] J. Huan, W. Wang, and J. Prins. "Efficient mining of frequent subgraphs in the presence of isomorphism." In *Third IEEE International Conference on Data Mining*, pp. 549–552. IEEE, 2003.

[IIP16] V. Ingalalli, D. Ienco, and P. Poncelet. "SuMGra: Querying multigraphs via efficient indexing." In *Database and Expert Systems Applications*, pp. 387–401. Springer, 2016.

[JHW19] H. Jin, X. He, Y. Wang, H. Li, and A. L. Bertozzi. "Noisy subgraph isomorphisms on multiplex networks." In *2019 IEEE International Conference on Big Data*, pp. 4899–4905. IEEE, 2019.

[JM18] A. Jüttner and P. Madarasi. "VF2++—an improved subgraph isomorphism algorithm." *Discrete Applied Mathematics*, **242**:69–81, 2018.

[JV87] R. Jonker and A. Volgenant. "A shortest augmenting path algorithm for dense and sparse linear assignment problems." *Computing*, **38**:325–340, 1987.

[JZ13] R. Johnson and T. Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pp. 315–323. Curran Associates, 2013.

[KAB14] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. "Multilayer networks." *Journal of Complex Networks*, **2**(3):203–271, 2014.

[Kac37] M. S. Kaczmarz. "Angenäherte auflösung von systemen linearer gleichungen." *Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques*, **35**:355–357, 1937.

[Kem81] A. Kemp. "Efficient generation of logarithmically distributed pseudo-random variables." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **30**(3):249–253, 1981.

[KF18] A. Katharopoulos and F. Fleuret. "Not all samples are created equal: Deep learning with importance sampling." In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2525–2534. PMLR, 2018.

[KMS16] L. Kotthoff, C. McCreesh, and C. Solnon. "Portfolios of subgraph isomorphism algorithms." In *Learning and Intelligent Optimization*, pp. 107–122. Springer, 2016.

[Knu00] D. E. Knuth. "Dancing links." *Millennial Perspectives in Computer Science*, pp. 187–214, 2000.

[KSD07] G. A. Korsah, A. Stentz, and M. B. Dias. "The dynamic Hungarian algorithm for the assignment problem with changing costs." Technical Report CMU-RI-TR-07-27, Carnegie Mellon University, 2007.

[KSG18] K. Karra, S. Swarup, and J. Graham. "An empirical assessment of the complexity and realism of synthetic social contact networks." In *2018 IEEE International Conference on Big Data*, pp. 3959–3967. IEEE, 2018.

[Kuh55] H. W. Kuhn. "The Hungarian method for the assignment problem." *Naval Research Logistics Quarterly*, **2**(1-2):83–97, 1955.

[KX19] A. Kopylov and J. Xu. "Filtering strategies for inexact subgraph matching on noisy multiplex networks." In *2019 IEEE International Conference on Big Data*, pp. 4906–4912. IEEE, 2019.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." *Nature*, **521**(7553):436–444, 2015.

[LH15] I. Loshchilov and F. Hutter. "Online batch selection for faster training of neural networks." *arXiv preprint arXiv:1511.06343*, 2015.

[LHN17] J. A. D. Loera, J. Haddock, and D. Needell. "A sampling Kaczmarz-Motzkin algorithm for linear feasibility." *SIAM Journal on Scientific Computing*, **39**(5):S66–S87, 2017.

[LL10] D. Leventhal and A. S. Lewis. "Randomized methods for linear constraints: convergence rates and conditioning." *Mathematics of Operations Research*, **35**(3):641–654, 2010.

[LT92] Z.-Q. Luo and P. Tseng. "On the convergence of the coordinate descent method for convex differentiable minimization." *Journal of Optimization Theory and Applications*, **72**(1):7–35, 1992.

[LV02] J. Larrosa and G. Valiente. "Constraint satisfaction algorithms for graph pattern matching." *Mathematical Structures in Computer Science*, **12**(4):403–422, 2002.

[LWS14] J. Liu, S. J. Wright, and S. Srikrishna. "An asynchronous parallel randomized Kaczmarz algorithm." *arXiv preprint arXiv:1401.4780*, 2014.

[LX15] Z. Lu and L. Xiao. "On the complexity analysis of randomized block-coordinate descent methods." *Mathematical Programming*, **152**(1–2):615–642, 2015.

[LZ17] Y. Liang and P. Zhao. "Similarity search in graph databases: A multi-layered indexing approach." In *2017 IEEE 33rd International Conference on Data Engineering*, pp. 783–794. IEEE, 2017.

[MBF20] G. Micale, V. Bonnici, A. Ferro, D. Shasha, R. Giugno, and A. Pulvirenti. "MultiRI: Fast subgraph matching in labeled multigraphs." *arXiv preprint arXiv:2003.11546*, 2020.

[McG79] J. J. McGregor. "Relational consistency algorithms and their application in finding subgraph and graph isomorphisms." *Information Sciences*, **19**(3):229–250, 1979.

[MNR15] A. Ma, D. Needell, and A. Ramdas. "Convergence properties of the randomized extended Gauss–Seidel and Kaczmarz methods." *SIAM Journal on Matrix Analysis and Applications*, **36**(4):1590–1604, 2015.

[MP15] C. McCreesh and P. Prosser. "A parallel, backjumping subgraph isomorphism algorithm using supplemental graphs." In *Principles and Practice of Constraint Programming*, pp. 295–312. Springer, 2015.

[MPS18] C. McCreesh, P. Prosser, C. Solnon, and J. Trimble. "When subgraph isomorphism is really hard, and why this matters for graph databases." *Journal of Artificial Intelligence Research*, **61**(1):723–759, 2018.

[MR11] M. Magnani and L. Rossi. "The ML-model for multi-layer social networks." In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pp. 5–12. IEEE, 2011.

[MS54] T. S. Motzkin and I. J. Schoenberg. "The relaxation method for linear inequalities." *Canadian Journal of Mathematics*, **6**:393–404, 1954.

[MTC20] J. D. Moorman, T. K. Tu, Q. Chen, D. Yang, and X. He. "jdmoorman/uclasm: v0.2.0." Zenodo. `https://doi.org/10.5281/zenodo.4052353`, 2020.

[MTC21] J. D. Moorman, T. Tu, Q. Chen, X. He, and A. L. Bertozzi. "Subgraph matching on multiplex networks." *IEEE Transactions on Network Science and Engineering*, 2021. Early Access.

[MTM21] J. D. Moorman, T. K. Tu, D. Molitor, and D. Needell. "Randomized Kaczmarz with averaging." *BIT Numerical Mathematics*, **61**(1):337–359, 2021.

[Mun57] J. Munkres. "Algorithms for the assignment and transportation problems." *Journal of the Society for Industrial and Applied Mathematics*, **5**(1):32–38, 1957.

[Nat01] F. Natterer. *The Mathematics of Computerized Tomography*. SIAM, 2001.

[Nec19] I. Necoara. "Faster randomized block Kaczmarz algorithms." *SIAM Journal on Matrix Analysis and Applications*, **40**(4):1425–1452, 2019.

[Nee10] D. Needell. "Randomized Kaczmarz solver for noisy linear systems." *BIT Numerical Mathematics*, **50**(2):395–403, 2010.

[Nes12] Y. Nesterov. "Efficiency of coordinate descent methods on huge-scale optimization problems." *SIAM Journal on Optimization*, **22**(2):341–362, 2012.

[NNG17] I. Necoara, Y. Nesterov, and F. Glineur. "Random block coordinate descent methods for linearly constrained optimization over networks." *Journal of Optimization Theory and Applications*, **173**(1):227–254, 2017.

[Nora] P. Norvig. "pytudes GitHub repository." `https://github.com/norvig/pytudes`, a. Accessed: 2021-05-30.

[Norb] P. Norvig. "Solving every Sudoku puzzle." `http://www.norvig.com/sudoku.html`, b. Accessed: 2021-05-30.

[NRR11] F. Niu, B. Recht, C. Ré, and S. J. Wright. "HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent." In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 693—701. Curran Associates, 2011.

[NS17a] C. Nabti and H. Seba. "Compact neighborhood index for subgraph queries in massive graphs." *arXiv preprint arXiv:1703.05547*, 2017a.

[NS17b] Y. Nesterov and S. U. Stich. "Efficiency of the accelerated coordinate descent method on structured optimization problems." *SIAM Journal on Optimization*, **27**(1):110–123, 2017b.

[NSL15] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke. "Coordinate descent converges faster with the Gauss–Southwell rule than random selection." In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1632–1641. PMLR, 2015.

[NSL16] J. Nutini, B. Sepehry, I. Laradji, M. Schmidt, H. Koepke, and A. Virani. "Convergence rates for greedy Kaczmarz algorithms, and faster randomized Kaczmarz rules using the orthogonality graph." In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, pp. 547–556. AUAI Press, 2016.

[NSW16] D. Needell, N. Srebro, and R. Ward. "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm." *Mathematical Programming*, **155**:549–573, 2016.

[NT14] D. Needell and J. A. Tropp. "Paved with good intentions: Analysis of a randomized block Kaczmarz method." *Linear Algebra and its Applications*, **441**:199–221, 2014.

[NW13] D. Needell and R. Ward. "Two-subspace projection method for coherent overdetermined systems." *Journal of Fourier Analysis and Applications*, **19**(2):256–269, 2013.

[NW17] D. Needell and R. Ward. "Batched stochastic gradient descent with weighted sampling." In *Approximation Theory XV: San Antonio 2016*, volume 201 of *Springer Proceedings in Mathematics & Statistics*, pp. 279–306. Springer, 2017.

[NYG19] T. Nguyen, D. Yang, Y. Ge, H. Li, and A. L. Bertozzi. "Applications of structural equivalence to subgraph isomorphism on multichannel multigraphs." In *2019 IEEE International Conference on Big Data*, pp. 4913–4920. IEEE, 2019.

[NZZ15] D. Needell, R. Zhao, and A. Zouzias. "Randomized block Kaczmarz method with projection for solving least squares." *Linear Algebra and its Applications*, **484**:322–343, 2015.

[OAL16] A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. Dokania, and S. Lacoste-Julien. "Minding the gaps for block Frank-Wolfe optimization of structured SVMs." In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 593–602. PMLR, 2016.

[PCJ17] D. Perekrestenko, V. Cevher, and M. Jaggi. "Faster coordinate descent via adaptive importance sampling." In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 869–877. PMLR, 2017.

[PJM19] V. Patel, M. Jahangoshahi, and D. A. Maldonado. "An implicit representation and iterative solution of randomly sketched linear systems." *arXiv preprint arXiv:1904.11919*, 2019.

[Pop99] C. Popa. "Characterization of the solutions set of inconsistent least-squares problems by an extended Kaczmarz algorithm." *Korean Journal of Computational and Applied Mathematics*, **6**(1):51–64, 1999.

[PP16] S. Petra and C. Popa. "Single projection Kaczmarz extended algorithms." *Numerical Algorithms*, **73**(3):791–806, 2016.

[PPP17] S. Pilosof, M. A. Porter, M. Pascual, and S. Kéfi. "The multilayer nature of ecological networks." *Nature Ecology & Evolution*, **1**(0101), 2017.

[Pro] "Project Euler problem 96: Su Doku." `https://projecteuler.net/problem=96`. Accessed: 2021-05-30.

[PTT14] R. Pienta, A. Tamersoy, H. Tong, and D. H. Chau. "MAGE: Matching approximate patterns in richly-attributed graphs." In *2014 IEEE International Conference on Big Data*, pp. 585–590. IEEE, 2014.

[RDM08] L. Rigutini, M. Diligenti, M. Maggini, and M. Gori. "A fully automatic crossword generator." In *2008 7th International Conference on Machine Learning and Applications*, pp. 362–367. IEEE, 2008.

[Reg94] J. Régin. "A filtering algorithm for constraints of difference in CSPs." In *Proceedings of the 12th National Conference on Artificial Intelligence*, volume 1, pp. 362–367. AAAI, 1994.

[RKR17] T. Reza, C. Klymko, M. Ripeanu, G. Sanders, and R. Pearce. "Towards practical and robust labeled pattern matching in trillion-edge graphs." In *2017 IEEE International Conference on Cluster Computing*, pp. 1–12. IEEE, 2017.

[RM51] H. Robbins and S. Monro. "A stochastic approximation method." *The Annals of Mathematical Statistics*, **22**(3):400–407, 1951.

[RT14] P. Richtárik and M. Takáč. "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function." *Mathematical Programming*, **144**(1):1–38, 2014.

[RT16] P. Richtárik and M. Takáč. "Distributed coordinate descent method for learning with big data." *Journal of Machine Learning Research*, **17**(75):1–25, 2016.

[RT20] P. Richtárik and M. Takáč. "Stochastic reformulations of linear systems: Algorithms and convergence theory." *SIAM Journal on Matrix Analysis and Applications*, **41**(2):487–524, 2020.

[RW15] X. Ren and J. Wang. "Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs." *Proceedings of the VLDB Endowment*, **8**(5):617–628, 2015.

[Sai79] L. D. Sailer. "Structural equivalence: Meaning and definition, computation and application." *Social Networks*, **1**(1):73–90, 1978–1979.

[SF83] A. Sanfeliu and K. Fu. "A distance measure between attributed relational graphs for pattern recognition." *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-13**(3):353–362, 1983.

[Sim05] H. Simonis. "Sudoku as a constraint problem." In *Proceedings of the 4th International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, pp. 13–27, 2005.

[SL19] S. Sun and X. Luo. "Scaling up subgraph query processing with efficient subgraph matching." In *2019 IEEE 35th International Conference on Data Engineering*, pp. 220–231. IEEE, 2019.

[Sol10] C. Solnon. "AllDifferent-based filtering for subgraph isomorphism." *Artificial Intelligence*, **174**(12–13):850–864, 2010.

[SPP20] D. Sussman, Y. Park, C. E. Priebe, and V. Lyzinski. "Matched filters for noisy induced subgraph detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **42**(11):2887–2900, 2020.

[SV09] T. Strohmer and R. Vershynin. "A randomized Kaczmarz algorithm with exponential convergence." *Journal of Fourier Analysis and Applications*, **15**(2):262–278, 2009.

[SW05] T. Schank and D. Wagner. "Finding, counting and listing all triangles in large graphs, an experimental study." In *Experimental and Efficient Algorithms*, pp. 606–609. Springer, 2005.

[SWW12] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li. "Efficient subgraph matching on billion node graphs." *Proceedings of the VLDB Endowment*, **5**(9):788—799, 2012.

[SZL08] H. Shang, Y. Zhang, X. Lin, and J. X. Yu. "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism." *Proceedings of the VLDB Endowment*, **1**(1):364–375, 2008.

[TMY20] T. K. Tu, J. D. Moorman, D. Yang, Q. Chen, and A. L. Bertozzi. "Inexact attributed subgraph matching." In *2020 IEEE International Conference on Big Data*, pp. 2575–2582. IEEE, 2020.

[Tse90] P. Tseng. "Dual ascent methods for problems with strictly convex costs and linear constraints: A unified approach." *SIAM Journal on Control and Optimization*, **28**(1):214–242, 1990.

[Ull76] J. R. Ullmann. "An algorithm for subgraph isomorphism." *Journal of the ACM*, **23**(1):31–42, 1976.

[Ull10] J. R. Ullmann. "Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism." *Journal of Experimental Algorithmics*, **15**(1.6), 2010.

[Ver79] L. M. Verbrugge. "Multiplexity in adult friendships." *Social Forces*, **57**(4):1286–1309, 1979.

[Wal74] A. J. Walker. "New fast method for generating discrete random numbers with arbitrary frequency distributions." *Electronics Letters*, **10**(8):127–128, 1974.

[XTL18] J. Xu, H. Tong, T. Lu, J. He, and N. Bliss. "GTA^3 2018: Workshop on graph techniques for adversarial activity analytics." In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, p. 803. ACM, 2018.

[XY15] Y. Xu and W. Yin. "Block stochastic gradient iteration for convex and nonconvex optimization." *SIAM Journal on Optimization*, **25**(3):1686–1716, 2015.

[XZ02] J. Xu and L. Zikatanov. "The method of alternating projections and the method of subspace corrections in Hilbert space." *Journal of the American Mathematical Society*, **15**(3):573–597, 2002.

[YG13] L. Yartseva and M. Grossglauser. "On the performance of percolation graph matching." In *Proceedings of the First ACM Conference on Online Social Networks*, pp. 119–130. ACM, 2013.

[YH02] X. Yan and J. Han. "gspan: Graph-based substructure pattern mining." In *2002 IEEE International Conference on Data Mining*, pp. 721–724. IEEE, 2002.

[ZDS10] S. Zampelli, Y. Deville, and C. Solnon. "Solving subgraph isomorphism problems with constraint programming." *Constraints*, **15**:327–353, 2010.

[ZF13] A. Zouzias and N. M. Freris. "Randomized extended Kaczmarz for solving least-squares." *SIAM Journal on Matrix Analysis and Applications*, **34**(2):773–793, 2013.

[ZH10] P. Zhao and J. Han. "On graph query optimization in large networks." *Proceedings of the VLDB Endowment*, **3**(1–2):340–351, 2010.

[ZL17] M. Zitnik and J. Leskovec. "Predicting multicellular function through multi-layer tissue networks." *Bioinformatics*, **33**(14):190–198, 2017.

[ZLY09] S. Zhang, S. Li, and J. Yang. "GADDI: distance index based subgraph matching in biological networks." In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 192–203. ACM, 2009.

[ZMC11] L. Zou, J. Mo, L. Chen, M. T. Özsu, and D. Zhao. "gStore: answering SPARQL queries via subgraph matching." *Proceedings of the VLDB Endowment*, **4**(8):482–493, 2011.

[ZZ15] P. Zhao and T. Zhang. "Stochastic optimization with importance sampling for regularized loss minimization." In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1–9. PMLR, 2015.