Non-Convex Optimization and Applications to Bilinear Programming and Super-Resolution Imaging

by

Jessica Gronski

B.S., University of Colorado, Colorado Springs, 2014

M.S., University of Colorado, Boulder, 2017

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2019

This thesis entitled:
Non-Convex Optimization and Applications to Bilinear Programming and Super-Resolution Imaging
written by Jessica Gronski
has been approved for the Department of Applied Mathematics

Prof. Stephen Becker

Prof. Sriram Sankaranarayanan

Prof. Carol Cogswell

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Gronski, Jessica (Ph.D., Applied Mathematics)

Non-Convex Optimization and Applications to Bilinear Programming and Super-Resolution Imaging

Thesis directed by Prof. Stephen Becker

Bilinear programs and Phase Retrieval are two instances of nonconvex problems that arise in engineering and physical applications, and both occur with their fundamental difficulties. In this thesis, we consider various methods and algorithms for tackling these challenging problems and discuss their effectiveness.

Bilinear programs (BLPs) are ubiquitous in engineering applications, economics, and operations research, and have a natural encoding to quadratic programs. They appear in the study of Lyapunov functions used to deduce the stability of solutions to differential equations describing dynamical systems. For multivariate dynamical systems, the problem formulation for computing an appropriate Lyapunov function is a BLP. In electric power systems engineering, one of the most practically important and well-researched subfields of constrained nonlinear optimization is Optimal Power Flow wherein one attempts to optimize an electric power system subject to physical constraints imposed by electrical laws and engineering limits, which can be naturally formulated as a quadratic program. In a recent publication [GBSBS18], we studied the relationship between data flow constraints for numerical domains such as polyhedra and bilinear constraints.

The problem of recovering an image from its Fourier modulus, or intensity, measurements emerges in many physical and engineering applications. The problem is known as Fourier phase retrieval wherein one attempts to recover the phase information of a signal in order to accurately reconstruct it from estimated intensity measurements by applying the inverse Fourier transform. The problem of recovering phase information from a set of measurements can be formulated as a quadratic program. This problem is well-studied but still presents many challenges. The resolution of an optical device is defined as the smallest distance between two objects such that the two objects

can still be recognized as separate entities. Due to the physics of diffraction, and the way that light bends around an obstacle, the resolving power of an optical system is limited. This limit, known as the diffraction limit, was first introduced by Ernst Abbe in 1873. Obtaining the complete phase information would enable one to perfectly reconstruct an image; however, the problem is severely ill-posed and the leads to a specialized type of quadratic program, known as super-resolution imaging, wherein one attempts to learn phase information beyond the limits of diffraction and the limitations imposed by the imaging device.

Dedication

For my mom.

Acknowledgements

I'd like to express my deepest gratitude to Stephen Becker. The insight and knowledge I've gained from working with him is invaluable. Without his patience, support, and encouragement this thesis would not have been possible.

I am grateful to Sriram Sankaranarayanan for his continued support and for introducing me to an exciting, and challenging field.

I would like to extend my sincere thanks to my committee. Their suggestions and guidance have helped shape this thesis, and for that, I am truly appreciative.

Finally, I'm indebted to the close relationships I've developed over the years. Graduate school wouldn't have been as amazing without the laughter and company of my peers.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

In this chapter, we give a brief overview of optimization utilizing Boyd and Vandenberge's *Convex Optimization*, and then discuss the outline of this thesis along with our contributions.

1.1 Preliminaries

There are many ways to categorize optimization problems based on their structure and tractability. One of the most common classifications is convex versus nonconvex problems. Convex functions have the property that all local optima are also global, making them tractable. Boyd and Vandenberghe [BV04] give a thorough overview of convex functions and the algorithms employed to solve them. In what follows, we explain some of the key features of convex optimization and its role in making seemingly intractable problems more tractable.

Definition 1.1.1. A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is convex if its effective domain, $S = \{\vec{x} \in \mathbb{R}^n | f(\vec{x}) < \infty\}$, is a convex set and

$$f(\lambda\vec{x} + (1-\lambda)\vec{y}) \leq \lambda f(\vec{x}) + (1-\lambda) f(\vec{y})$$

for all $\vec{x}, \vec{y} \in \text{dom}(f)$ and $0 \leq \lambda \leq 1$.

Being a convex set means for all points $\vec{x}, \vec{y} \in S$ and $0 \leq \lambda \leq 1$ we have $\lambda\vec{x} + (1-\lambda)\vec{y} \in S$; that is, a set is convex if for any two points $\vec{x}, \vec{y} \in S$ the line segment connecting $\vec{x}$ and $\vec{y}$ is contained in S. From a geometric standpoint, the functional inequality means that any chord from $\vec{x}$ to $\vec{y}$ lies above the graph of f. A function is strictly convex if the inequality is strict whenever

$\vec{x} \neq \vec{y}$ and $0 < \lambda < 1$. There is a close connection between convex functions and convex sets: a function f is convex if and only if the epigraph of f, which is a subset of $\mathbb{R}^n \times \mathbb{R}$ consisting of all points $(\vec{x}, t)$ with $f(\vec{x}) \leq t$, is a convex set. Affine functions are convex since the above inequality is always an equality. Furthermore, we can check whether a function is convex by restricting it to any line bisecting the domain and verifying that that function is indeed convex. Convex functions also benefit from the fact that they are continuous on the relative interior of their domain, with the possibility of discontinuities on their relative boundary. Other convenient properties of convex functions include:

- If $f, g : \mathbb{R}^n \to \mathbb{R}$ are two convex functions, then their sum $f + g$ is also convex.
- If $f : \mathbb{R}^n \to \mathbb{R}$ is convex and $\lambda \geq 0$, then λf is convex.
- If f and g are convex functions, then the function $h(\vec{x}) := \max\{f(\vec{x}), g(\vec{x})\}$ is also convex.

In addition, a function h is called concave if and only if $-h$ is convex. Therefore, any function that is both convex and concave must be affine. There is no explicit concept of a concave set; however, we say a set is concave if its complement is convex.

A variable $\vec{x}$ is a global minimum of f if

$$f(\vec{x}) \leq f(\vec{y})$$

holds for any $\vec{y}$ in the domain of f, and a local minimum if it holds for every $\vec{y}$ within some neighborhood of $\vec{x}$. While local optimality does not necessarily imply global optimality, global optimality always implies local. However, for convex functions, the two coincide. From the geometric interpretation, if a convex function had a local minimum that was not also global, then any chord connecting the two values could not lie entirely above the graph of f, thus contradicting the definition of a convex function. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. Then f is convex if and only if for every $\vec{x}, \vec{y} \in \mathbb{R}^n$

$$f(\vec{x}) + \nabla f(\vec{x})^T (\vec{y} - \vec{x}) \leq f(\vec{y})$$

holds and $\text{dom}(f)$ is a convex set.

Notice that $f(\vec{x}) + \nabla f(\vec{x})^T(\vec{y} - \vec{x})$ is an affine function of $\vec{y}$. Thus, the first-order Taylor approximation of f near $\vec{x}$ is a global underestimator of f. Furthermore, recall that $\nabla f(\vec{x}^*) = 0$ is a necessary condition for $\vec{x}^*$ to be a local minimizer of f. If f is a convex function and $\vec{x}^*$ satisfies $\nabla f(\vec{x}^*) = 0$, then

$$f(\vec{x}^*) = f(\vec{x}^*) + \nabla f(\vec{x}^*)^T(\vec{y} - \vec{x}^*) \leq f(\vec{y}), \tag{1.1}$$

and so $\vec{x}^*$ is in fact a global minimizer of f.

The following property of convex functions requires us to define an important property of square matrices.

Definition 1.1.2. A symmetric $n \times n$ real matrix A is said to be positive definite if $\vec{x}^T A\vec{x} > 0$, positive semidefinite if $\vec{x}^T A\vec{x} \geq 0$, and negative definite if $\vec{x}^T A\vec{x} < 0$ for every non-zero vector $\vec{x} \in \mathbb{R}^n$.

Another way to characterize the definiteness of a matrix is to inspect its eigenvalues. If all eigenvalues of A are positive, then the matrix A is said to be positive definite, and so forth. If some of the eigenvalues of A are positive and some are negative, then A is called indefinite.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable function. Then f is convex if and only if the Hessian $\nabla^2 f(\vec{x})$ is positive definite for all $\vec{x} \in \text{dom}(f)$ and $\text{dom}(f)$ is a convex set. The proof of this, and the previous claim, can be found in [BV04]. Geometrically, the Hessian being positive definite for all $\vec{x}$ in the domain of f means that the local curvature at $\vec{x}$ is positive, or upward.

Definition 1.1.3. The convex hull of a set S, denoted $\text{conv}(S)$, is the smallest convex set containing S, i.e., it is the intersection of all convex sets containing S.

Some elementary convex sets include:

- Hyperplanes: $\vec{a}^T\vec{x} = b$
- Half-spaces: $\vec{a}^T\vec{x} \leq b$

- Affine sets: $A\vec{x} = \vec{b}$
- Polyhedra sets: $A\vec{x} \leq \vec{b}$
- All positive (semi)definite matrices form a convex cone $X \succ 0$ ($X \succeq 0$)

The most general form of a continuous optimization problems is

$$\begin{aligned} \min_{\vec{x}} \quad & f_0(\vec{x}) \\ \text{subject to} \quad & f_i(\vec{x}) \leq 0, \quad i \in \mathcal{I} \\ & h_i(\vec{x}) = 0, \quad i \in \mathcal{E} \end{aligned} \tag{1.2}$$

where $f_0(\vec{x})$ is typically referred to as the objective or cost function and $\vec{x} \in \mathbb{R}^n$ are the program variables. $\mathcal{I}$ and $\mathcal{E}$ are the inequality and equality index sets for the inequality, $f_i : \mathbb{R}^n \to \mathbb{R}$, and equality, $h_i : \mathbb{R}^n \to \mathbb{R}$, constraints, respectively. If $\mathcal{I} = \mathcal{E} = \emptyset$, i.e., the empty set, then we say the problem is unconstrained. A point $\vec{x}$ in the domain of the problem if called feasible if all constraints $f_i(\vec{x})$ and $h_i(\vec{x})$ are satisfied at $\vec{x}$. If at least one point is feasible, then the problem is called feasible; otherwise, it is called infeasible. The set of all feasible points is called the feasible set, or constraint set, of the problem. If all equality constraints are affine, i.e., $h_i(\vec{x}) = \vec{a}_i^T\vec{x} - b_i$ for all $i \in \mathcal{E}$, and each of the functions $f_0(\vec{x}), f_i(\vec{x})$, for $i \in \mathcal{I}$, is convex, then (1.2) is called a convex optimization problem.

The optimal value, p^*, of (1.2) is

$$p^* = \inf\{\ f_0(\vec{x}) \mid f_i(\vec{x}) \leq 0,\ i \in \mathcal{I},\ h_i(\vec{x}) = 0,\ i \in \mathcal{E}\}.$$

If $p^* = -\infty$ we say the problem is unbounded, and we use the convention $p^* = \infty$ if the problem is infeasible. Oftentimes, if it is unclear whether the given problem is feasible or not, one may solve a feasibility problem wherein the objective function is identically zero

$$\begin{aligned} \text{find} \quad & \vec{x} \\ \text{subject to} \quad & f_i(\vec{x}) \leq 0, \quad i \in \mathcal{I} \\ & h_i(\vec{x}) = 0, \quad i \in \mathcal{E}. \end{aligned} \tag{1.3}$$

In this case, $p^* = 0$ if the constraints are consistent, else, $p^* = \infty$.

There are various transformations that allow us to rewrite the original problem as an equivalent optimization problem. Two problems are called equivalent if we can obtain the solution of one problem from the solution of the other. This may be accomplished by performing a one-to-one change of variables or by composing the original function with new functions satisfying appropriate conditions. We will see later that when there are inequality constraints, a useful strategy for transforming the inequality constraints into equality constraints is to introduce slack variables s_i. This transformation will be key when implementing optimization algorithms that naturally handle equality constraints, such as penalty methods. Note that $f_i(\vec{x}) \leq 0$ if and only if $f_i(\vec{x}) + s_i = 0$ for some $s_i \geq 0$. This way, we are able to transform the original problem (1.2) into the equivalent

$$\begin{aligned} \min_{\vec{x},\vec{s}} \quad & f_0(\vec{x}) \\ \text{subject to} \quad & f_i(\vec{x}) + s_i = 0, \quad i \in \mathcal{I} \\ & h_i(\vec{x}) = 0, \qquad i \in \mathcal{E} \\ & s_i \geq 0, \qquad i \in \mathcal{I} \end{aligned} \tag{1.4}$$

where $\vec{x} \in \mathbb{R}^n$ and $\vec{s} \in \mathbb{R}^{|\mathcal{I}|}$, $|\mathcal{I}|$ is the cardinality of $\mathcal{I}$. The new variables s_i are called slack variables, and since the non-negative orthant is a convex set, if (1.2) is a convex program then so is (1.4).

In optimization, there is an important concept known as duality. Duality provides us the means to view an optimization problem from either of two perspectives: the primal problem and the dual problem, and solving the dual problem can provide some insight on the primal problem. For instance, if we are trying to minimize (maximize) some objective subject to a set of constraints, then the optimal value of the dual program, say d^*, provides a lower (upper) bound for the optimal value of the primal, p^*. Their difference, $p^* - d^*$, is referred to as the duality gap, and for convex programs, the duality gap is zero provided the relative interior of the feasible set is non-empty. Let

$$D = \bigcap_{i=0}^{|\mathcal{I}|} \text{dom}(f_i) \bigcap_{i=1}^{|\mathcal{E}|} \text{dom}(h_i)$$

denote the domain of (1.2). Given the optimization problem (1.2), with the domain $D \subset \mathbb{R}^n$ having non-empty interior, the Lagrangian function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{E}|} \to \mathbb{R}$ is defined as

$$\mathcal{L}(\vec{x}, \vec{\lambda}, \nu) = f_0(\vec{x}) + \sum_{i=1}^{|\mathcal{I}|} \lambda_i f_i(\vec{x}) + \sum_{i=1}^{|\mathcal{E}|} \nu_i h_i(\vec{x}). \tag{1.5}$$

$\vec{\lambda}$ and $\vec{\nu}$ are called the dual variables, or Lagrange multipliers associated with the problem. The Lagrangian dual function $g : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{E}|} \to \mathbb{R}$ is defined as

$$\begin{aligned} g(\lambda, \nu) &= \inf_{\vec{x} \in D} \mathcal{L}(\vec{x}, \vec{\lambda}, \nu) \\ &= \inf_{\vec{x} \in D} \left(f_0(\vec{x}) + \sum_{i=1}^{|\mathcal{I}|} \lambda_i f_i(\vec{x}) + \sum_{i=1}^{|\mathcal{E}|} \nu_i h_i(\vec{x}) \right). \end{aligned}$$

The idea is similar to that of the Lagrangian function in multivariate calculus; however, now we allow for inequality constraints. All constraints are taken into account by augmenting the objective with a weighted combination of the constraints. Even when the original problem is not convex, the dual function is always concave. If $\vec{\lambda} \geq 0$ component-wise, preserving the original inequality, then $g(\vec{\lambda}, \nu) \leq p^*$. This is immediately apparent by considering any feasible point $\overline{x}$ and $\vec{\lambda} \geq 0$,

$$f_0(\overline{x}) \geq \mathcal{L}(\overline{x}, \vec{\lambda}, \nu) \geq \inf_{\vec{x} \in D} \mathcal{L}(\vec{x}, \vec{\lambda}, \nu) = g(\vec{\lambda}, \nu).$$

The Lagrangian dual problem associated with (1.2) is

$$\begin{aligned} \max_{\vec{\lambda}, \nu} \quad & g(\vec{\lambda}, \nu) \\ \text{subject to} \quad & \lambda_i \geq 0, \quad i \in \mathcal{I} \end{aligned} \tag{1.6}$$

In section 3.3 we introduce a novel penalty method approach to solving bilinear programs using the augmented Lagrangian framework.

Strong duality implies $d^* = p^*$ and, in general, does not hold. However, weak duality, $d^* \leq p^*$ always holds and can be used to find non-trivial lower bounds for difficult problems. In this context, dual feasibility describes a pair $(\vec{\lambda}, \nu)$ such that $\vec{\lambda} \succeq 0$ where we use $\vec{x} \succeq 0$ to denote $\vec{x}$ is element-wise non-negative, and $g(\vec{\lambda}, \nu) > -\infty$. A dual feasible point provides a certificate that $p^* \geq g(\vec{\lambda}, \nu)$.

Without knowing the exact value of p^*, we can still quantify the suboptimality of a primal feasible point using dual feasible points. For a primal feasible $\vec{x}$

$$f_0(\vec{x}) - p^* \leq f_0(\vec{x}) - g(\vec{\lambda}, \nu). \tag{1.7}$$

Thus, $\vec{x}$ can be thought of as ϵ-suboptimal with $\epsilon = f_0(\vec{x}) - g(\vec{\lambda}, \nu)$. Furthermore, (1.6) is a convex optimization problem since we are tasked with maximizing a concave function. Therefore, any local optimum is necessarily a global one. In section 3.4 we introduce a semidefinite programming relaxation using duality to solving quadratic programs that can be directly applied to solving BLPs, and under certain conditions, we have strong duality.

Definition 1.1.4. (Complementary Slackness) Assume strong duality holds for (1.2). Let $\vec{x}^*$ be primal optimal and $(\vec{\lambda}^*, \nu^*)$ be dual optimal, then

$$\begin{aligned} f_0(\vec{x}^*) = g(\vec{\lambda}^*, \nu^*) &= \inf_{\vec{x} \in D} \left(f_0(\vec{x}) + \sum_{i=1}^{|\mathcal{I}|} \lambda_i f_i(\vec{x}) + \sum_{i=1}^{|\mathcal{E}|} \nu_i h_i(\vec{x}) \right) \\ &\leq f_0(\vec{x}^*) + \sum_{i=1}^{|\mathcal{I}|} \lambda_i^* f_i(\vec{x}^*) + \sum_{i=1}^{|\mathcal{E}|} \nu_i^* h_i(\vec{x}^*) \\ &\leq f_0(\vec{x}^*). \end{aligned}$$

Consequently, the two inequalities must hold with equality and so $\lambda_i^* f_i(\vec{x}^*) = 0$ for $i \in \mathcal{I}$. Thus,

$$\lambda_i^* > 0 \implies f_i(\vec{x}^*) = 0, \qquad f_i(\vec{x}^*) < 0 \implies \lambda_i^* = 0, \tag{1.8}$$

a property known as complementary slackness.

Moreover, we say that a constraint is active at a feasible point $\vec{x}^*$ if $f_i(\vec{x}^*) = 0$, and inactive if $f_i(\vec{x}^*) < 0$. This means the ith optimal Lagrange multiplier is zero unless the ith constraint is active at the optimum.

The Karush-Kuhn-Tucker (KKT) conditions are first-order necessary conditions for a solution of an optimization problem to be optimal. To compute the KKT conditions, we require that each function $f_0(\vec{x})$, $f_i(\vec{x})$ for $i \in \mathcal{I}$, and $h_i(\vec{x})$ for $i \in \mathcal{E}$ be continuously differentiable at a point $\vec{x}^*$. They are a generalization of Lagrange multipliers, which specify conditions for equality constrained

optimization problems, i.e., $\mathcal{I} = \emptyset$ in (1.2). Let $(\vec{x}^*, \vec{\lambda}^*, \nu^*)$ be any primal and dual optimal points with zero duality gap. The Karush-Kuhn-Tucker conditions for optimality are

$$\begin{aligned} f_i(\vec{x}^*) &\leq 0, \quad i \in \mathcal{I} \\ h_i(\vec{x}^*) &= 0, \quad i \in \mathcal{E} \\ \lambda_i^* &\geq 0, \quad i \in \mathcal{I} \\ \lambda_i^* f_i(\vec{x}^*) &= 0, \quad i \in \mathcal{I} \\ \nabla f_0(\vec{x}^*) + \sum_{i=1}^{|\mathcal{I}|} \lambda_i^* \nabla f_i(\vec{x}^*) + \sum_{i=1}^{|\mathcal{E}|} \nu_i^* \nabla h_i(\vec{x}^*) &= 0. \end{aligned}$$

The first three conditions state that the point $(\vec{x}^*, \vec{\lambda}^*, \nu^*)$ must be primal and dual feasible. The fourth condition enforces complementary slackness between the dual variables λ_i^* and the set of inequality constraints, and the last statement describes stationarity at the optimal point. When the primal problem is convex, these conditions are sufficient for deducing optimality.

Theorem 1.1.1. *(Slater's Condition) Let the primal problem* (1.2) *be convex and bounded from below, i.e., $f_0(\vec{x}) > -\infty$ for $\vec{x} \in D$. Assume there exists an $\vec{x}_0$ that is strictly feasible, or satisfies the (non-affine) inequalities strictly, then a KKT vector (not-necessarily unique) exists.*

For convex programs, Slater's condition implies strong duality, and the dual optimum is attained. The set of KKT conditions generalize the optimality condition for unconstrained problems. In certain cases, it may be possible to solve the KKT conditions analytically. However, in general, there is no closed-form solution for solving them directly. Many optimization algorithms can be viewed as methods for solving the KKT system of equations and inequalities.

Definition 1.1.5. (Strong and Weak Alternatives) Two systems of inequalities (and equalities) are called weak alternatives if at most one of the two is feasible. Strong alternatives describe a system of inequalities (and equalities) in which exactly one of the two alternatives hold.

Lemma 1.1.2. *(Farkas' Lemma) Let $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$. Then exactly one of the two following statements hold:*

(1) There exists an $\vec{x} \in \mathbb{R}^n$ *such that* $A\vec{x} = \vec{b}$ *and* $\vec{x} \geq 0$.

(2) There exists a $\vec{y} \in \mathbb{R}^m$ *such that* $A^T\vec{y} \geq 0$ *and* $\vec{b}^T\vec{y} < 0$.

Farkas' lemma is an example of strong alternatives. Geometrically, the first statement states that there exists a set of non-negative coefficients such that $\vec{b}$ is in the range of A (or more specifically, $\vec{b}$ lies in the convex cone generated by the columns of A), while the second statement implies there exists a vector $\vec{y}$ such that $\vec{a}_i^T\vec{y} \geq 0$ for $i = 1, \ldots, n$ and $\vec{b}^T\vec{y} < 0$, meaning the cone of A and $\vec{b}$ are separated by a hyperplane going through the origin. There are several variants of Farkas' lemma, but the idea is always the same: if a set of axioms is inconsistent, then it can be refuted using the derivation rules [MG07]. We use Farkas' lemma in section 2.4 to derive a system of equations corresponding to data flow equations for template domains. This allows us to define the problem as a bilinear program and certify the existence of a solution.

1.2 Contributions and Organization of this Thesis

This thesis is organized into three chapters. Chapter 2 is the first part of a publication[1] where we studied the template polyhedral abstract domain using connections to bilinear optimization techniques. Specifically, data flow constraints for numerical domains such as polyhedra can be expressed in terms of bilinear constraints. In Chapter 3 we propose algorithms such as policy and strategy iteration for solving the bilinear constraints that arise from template polyhedra, wherein the desired invariants conform to a fixed template form. We empirically compare policy iteration with a variety of other approaches for bilinear programming. These approaches adapt well-known algorithms to the special case of bilinear programs as well as using off-the-shelf tools for nonlinear programming. Part of this chapter has been published and corresponds to the experimental portion of Chapter 2. Sections 3.4-3.6 describe results not included in the paper, but to the best of our knowledge are new approaches to solving BLPs, and we introduce a novel approach in section 3.7. The implementation details of these (and the previously considered) methods can be found in section

[1] J. Gronski, M. Ben Sassi, S. Becker, and S. Sankaranarayanan, Template Polyhedra and Bilinear Optimization, Formal Methods in System Design (2018).

3.8. Furthermore, in section 3.9 we examine the efficacy of each solver on randomly generated bilinear problems of varying degrees of complexity by comparing their results and runtime. These comparisons were not included in the aforementioned publication.

In Chapter 4 we discuss the Phase Retrieval problem for reconstructing an image from its intensity measurements. We give a brief overview of current approaches used to solve the problem, some of which are close variants to the methods described in Chapter 3. Super-resolution imaging is discussed in section 4.4 and we propose a novel approach to quantifying the effectiveness of super-resolution techniques in section 4.5.3. In section 4.5.2 we describe current methods for estimating the entropy and mutual information of a system that entail discretizing the parameter space. However, these approaches underestimate the entropy by including a negative bias. We motivate the necessity for a better means of judging current practices, and introduce the concept of using Monte Carlo methods for facilitating this for future work.

Chapter 2

Template Polyhedra and Bilinear Optimization[1]

In this chapter, we exploit the connections between inferring post-fixed points (inductive invariants) for numerical domains and the process of solving nonlinear constraints to provide a template polyhedral domain that can modify the templates on-the-fly as the analysis progresses. In a template abstract domain, we fix the left-hand side expressions of the invariant properties of interest and use abstract interpretation to compute valid right-hand side constants so that the resulting inequalities form an inductive invariant [SSM05]. Template domains generalize a host of popular, "weakly domains" such as intervals [CC76], octagons [Min01a, Min01b], octahedra [CC07], pentagons [LF08], linear templates [SSM05], and quadratic templates [AGG12]. These domains have been well studied and proven to be effective for proving safety of runtime assertions in software [GPBG08, BCC$^+$05, BCC$^+$03, DS07, Matb, VB04]. Template domains have given rise to specialized approaches such as policy iteration [CGG$^+$05, GGTZ07] for improving post-fixed points, and strategy iteration for computing the least fixed point [GS11, GS07].

Policy iteration starts from a known post-fixed point, and alternates between finding a "policy" that certifies the current solution versus finding the best solution under the current "policy". This approach was originally proposed by Costan et al. for the interval domain [CGG$^+$05] and generalized to arbitrary templates subsequently [GGTZ07]. Extensions have been proposed for quadratic templates [AGG12]. On the other hand, strategy iteration approach works in a bottom

[1] This chapter has been published:
J. Gronski, M. Ben Sassi, S. Becker, and S. Sankaranarayanan, Template Polyhedra and Bilinear Optimization, Formal Methods in System Design (2018).

up fashion starting from the bottom of the lattice and exploiting the "monotonicity" property in the dataflow equations for the template domain [GS07]. Specifically, the system of data flow equations are linearized around the current solution, and a fixed point of the linearized system is obtained as the next solution.

This analysis is divided into two parts. In this chapter, we exploit the connection between policy iteration approach and classic bilinear optimization problems to design approaches that can vary the template on-the-fly. This is done by adapting policy iteration, which is a variant of the popular alternating coordinate descent that has been used widely in the control systems and optimization communities [GB94]. Using this connection, we notice that the alternation between solutions and multipliers can be extended to update the templates on the fly, as the iteration proceeds. Significantly, the update to the templates can be made **property-directed** in a simple manner. By combining these observations, we arrive at a policy iteration approach that can start from initial, user-defined templates and update them on the fly. An implementation of the approach and evaluation over a set of small benchmarks shows that the approach of updating the policies on the fly is an effective solution to inferring appropriate templates in a property directed manner. However policy iteration is not guaranteed to converge to a globally optimal solution, which would correspond to the least fixed point solution in the abstract domain. In practice, the technique gets stuck in a local minimum, yielding a suboptimal solution.

In the following chapter, we empirically compare policy iteration approach against other related approaches to local and global optimization problems [BBC+08, coi16, Bel09, WB06, Sah17, TS05]. A result by Helton and Merino on more general biconvex programs suggests that the alternating minimization almost never converges to a local minimum (technically a solution satisfying the KKT conditions) [HM97]. Adjé et al. demonstrate an approach that computes an optimal solution for systems which are nonexpansive [AGG14]. However, the general applicability of this result is unclear.

Thus, given evidence that policy iteration (or alternating minimization) may not be a good method, we explore alternatives, and run numerical experiments in Section 3.10, even proposing

our own new variant. In addition to the solvers mentioned in our paper [GBSBS18], we include several other methods, including projected gradient descent and a novel graph-partition approach to alternating minimization, and compare the efficacy of each solver on randomly generated bilinear programs of varying structure. For these examples, we also compare the runtime of each approach. Our results suggest that, at least on our benchmark problems, alternating minimization is in fact by far the best method, even compared with global optimization solvers. Furthermore, our implementation uses floating point, not exact arithmetic, and still achieves acceptable accuracy. Finally, although we do not focus on runtime for our benchmark problems, we do note that alternating minimization, in floating point arithmetic, is also one of the fastest solvers. The conclusion is that although alternating minimization can have difficulties with saddle points, because it exploits the specific structure of the problem, it may still be the best choice in many practical cases.

2.1 Related Work

Colón et al. were the first to discover the connection between linear invariant synthesis problems and bilinear constraints through the use of Farkas lemma in linear programming [CSS03]. These constraints were solved using specialized quantifier elimination techniques, but restricted to small problems [Wei97]. Sankaranarayanan et al. explored the use of heuristic approaches to solve bilinear constraints [SSM04b]. These approaches were generalized by Cousot, as instances of **Lagrangian relaxations** [Cou05]. Additionally, Cousot's work uses numerical optimization tools to prove total correctness properties of programs. His approach relies on formulating the constraints as Linear or Bilinear Matrix inequalities (LMI/BMI). However, the use of numerical solvers requires rigorous symbolic verification of the results. Recent experiences reveal surprising pitfalls, including erroneous invariants obtained, even when the error tolerances are quite low [RVS16, SSCÁ16]. In fact, one of the advantages of policy iterations lies in the use of exact arithmetic LP solvers to avoid floating point errors. Other approaches to solving the resulting constraints have restricted the multiplier variables to finite domains, enabling linear arithmetic solvers [GSV08].

Template polyhedra and their generalization to support functions have proven useful for con-

structing reachable sets of linear and nonlinear hybrid systems [SDI08b, GG10, FLD$^+$11, CAS12]. The problem of inferring template directions has also been studied in this context. Many heuristics were proposed by Sankaranarayanan et al. in their paper on linear templates, including the use of expressions found in programs, "increasing"/ "decreasing" expressions, and preconditions of already added template expressions [SSM05]. However, none of these are guaranteed to be relevant to the property. Adjé et al. use the idea of Lyapunov-like functions to effectively infer templates that are shown to be effective in proving bounds on variables [AG15b].

The idea of updating templates on the fly was previously proposed by Ben Sassi et al. for analyzing the largest invariant region of a dynamical system [SGS14]. The approach searches for a polytope whose facets are transverse to the flow, failing which, the facet directions are adjusted and tested again. The approach to adjusting facets is based on a local sensitivity analysis to obtain the invariant region around an equilibrium (which facilitates basin of attraction analysis for dynamical systems). Compared to the present work, the differences include the treatment of multiple program locations and transitions, the use of policy iteration, and a property-directed approach that seeks to prove a property rather than find a largest invariant region.

Abraham et al. propose effective heuristics to guide the choice of directions for constructing reachable sets of linear hybrid systems [CE12]. Recently, Bogomolov et al. propose a counter-example guided approach for inferring facets of template polyhedra for hybrid systems reachability analysis [BFGH17]. The key differences include: (a) we are interested in computing a single polyhedron per location whereas flowpipe construction approaches use a disjunction of polytopes, and (b) we seek to compute time-unbounded invariants, whereas flowpipes are typically time bounded. Another interesting approach by Amato et al. uses principal component analysis (PCA) over concrete states reached by execution traces to design templates [APS10].

2.2 Motivating Example

Consider a simple system over two real-valued variables $(x_1, x_2) \in \mathbb{R}^2$, initialized to $(x_1, x_2) \in [-1, 1] \times [-1, 1]$. The system executes the following action

$$\textbf{if } (x_1, x_2) \in [-8, 8]^2 \textbf{ then } \left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := M \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right] \textbf{ else } \left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right]$$

wherein $M = \begin{pmatrix} 0.92 & 0.18 \\ 0.18 & 0.92 \end{pmatrix}$. Our goal is to prove that the set $U : \{(x_1, x_2) \mid x_2 - x_1 \geq 2.1\}$ is never reached by any execution of the system. In order to prove the property using a template domain, the user specifies a template matrix [SSM05, GGTZ07]:

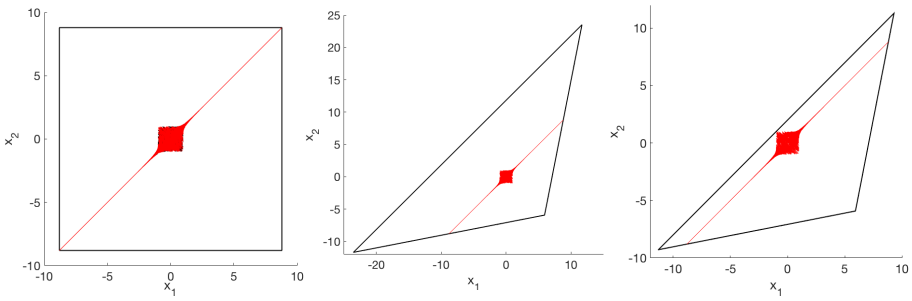
$$T : \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix}, \quad \begin{matrix} (* \ 1x_1 + 0x_2 \ *) \\ (* \ -1x_1 + 0x_2 \ *) \\ (* \ 0x_1 + 1x_2 \ *) \\ (* \ 0x_1 - 1x_2 \ *) \end{matrix}$$

wherein the rows represent the expressions $x_1, -x_1, x_2, -x_2$, respectively. The template domain analysis seeks to find an invariant of the form $T\vec{x} \leq \vec{c}$ by discovering the unknown constants $\vec{c}$ that represent the RHS of the template. For the example shown above, the best possible invariant is obtained as $\vec{c} : \begin{pmatrix} 8.8 & 8.8 & 8.8 & 8.8 \end{pmatrix}^T$, yielding the range $[-8.8, 8.8] \times [-8.8, 8.8]$ for (x_1, x_2). In fact, given our instance on using the template T, this is the best invariant possible (see Fig. 2.1(a) to verify this).

For this example, the policy iterative scheme presented in this chapter is successful in choosing a new template:

$$\hat{T} : \begin{pmatrix} -1 & 1 \\ 1 & -0.1957 \\ 0.1957 & -1 \\ -1 & 1 \end{pmatrix}, \quad \begin{matrix} (* \ -x_1 + x_2 \ *) \\ (* \ x_1 - 0.1957x_2 \ *) \\ (* \ 0.1957x_1 - x_2 \ *) \\ (* \ -x_1 + x_2 \ *) \end{matrix}$$

Figure 2.1: Invariants synthesized for the three steps of the policy iteration with property directed template modification. The simulation traces are shown in red. Note: each figure is drawn to a different scale.



Along with this policy, we compute a tighter invariant shown in Fig. 2.1**(c)**, that establishes the invariant $x_2 - x_1 \leq 2$, and thus proving U unreachable. We note that (a) the choice of templates is directed by the property, and (b) unlike the original policy iteration approach proposed by Gaubert et al. [GGTZ07], this approach does not guarantee that the iterates are strictly descending. In fact, the iterates obtained are often incomparable.

2.3 Preliminaries

Let $\mathbb{R}$ denote the set of real numbers and $\mathbb{R}_+ : \mathbb{R} \cup \pm\infty$ denote the extended reals with infinity. We first define the transition system model used throughout this chapter. Let X be a set of real-valued variables and $\Pi[X]$ represent a language of assertions over these variables, drawn from a suitable fragment of the first order logic over the reals. For any assertion $\varphi \in \Pi[X]$, we denote its corresponding set of models by $[\![\varphi]\!]$. For convenience, the set of variables X are arranged as a column vector, written as $\vec{x}$.

Definition 2.3.1 (Transition System)**.** A (numerical) **transition system** is a tuple $\langle X, \mathcal{L}, \mathcal{T}, \ell_0, \Theta \rangle$, wherein

(1) X : $\{x_1, \ldots, x_n\}$ represents a set of **real-valued** program variables,

(2) $\mathcal{L}$: $\{\ell_1, \ldots, \ell_m\}$ represents a set of program locations,

(3) $\mathcal{T}$: $\{\tau_1, \ldots, \tau_k\}$ represents a set of transitions, wherein each transition τ_i is a tuple $\langle \ell_i, m_i, \psi_i, g_i \rangle$, wherein,

(a) $\ell_i, m_i \in \mathcal{L}$ are the pre and the post locations, respectively.

(b) $\psi_i \in \Pi[X]$, an assertion over X, represents the guard of the transition.

(c) $g_i : \mathbb{R}^n \to \mathbb{R}^n$, an update function, represents the (simultaneous) assignment:

$(x_1, \ldots, x_n) := g_i(x_1, \ldots, x_n)$.

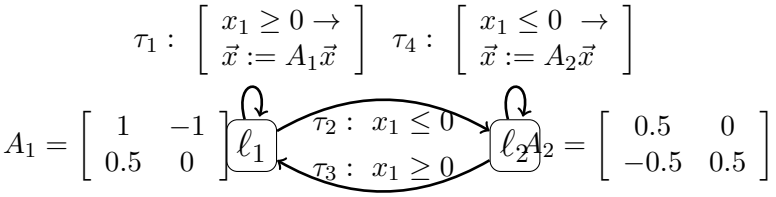
(4) ℓ_0 is the initial location, and $\Theta \in \Pi[X]$ is an assertion over X representing the initial valuations of the program variables.

A state of the transition system is a tuple $\langle \ell, \vec{x} \rangle$ wherein $\ell \in \mathcal{L}$ is the control location and $\vec{x} \in \mathbb{R}^n$ represents a set of valuations for the program variable. Given a transition system, its executions are a finite/infinite sequence of states:

$$(\ell_0, \vec{x}_0) \xrightarrow{\tau_1} (\ell_1, \vec{x}_1) \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_i} (\ell_i, \vec{x}_i) \cdots ,$$

such that: (a) ℓ_0 is the initial location and $\vec{x}_0 \in [\![\Theta]\!]$; (b) ℓ_{i-1}, ℓ_i are the pre/post locations (respectively) of the transition τ_i for all $i \geq 1$; (c) $\vec{x}_{i-1} \in [\![\psi_i]\!]$ for all $i \geq 1$ wherein ψ_i is the guard corresponding to the transition τ_i; and (d) $\vec{x}_i = g_i(\vec{x}_{i-1})$ for all $i \geq 1$, wherein g_i is the update function for τ_i.

Figure 2.2: Example of a transition system with two variables x_1, x_2, two locations ℓ_1, ℓ_2 and four transitions shown as arrows.



Example Figure 2.2 shows an example of a transition system with $X : \{x_1, x_2\}$, $\mathcal{L} : \{\ell_1, \ell_2\}$ and $\mathcal{T} : \{\tau_1, \tau_2, \tau_3, \tau_4\}$. The guards and updates of the transitions are as shown in Fig. 2.2. The identity

update $\vec{x} := \vec{x}$ is not shown, however. The initial location is ℓ_1 and the initial condition on $\vec{x}$ is $(x_1, x_2) \in [0.5, 1.5] \times [0.5, 1]$.

A state $(\ell, \vec{x})$ is reachable if there is an execution that reaches the state.

For this exposition, we study **linear transition systems**. A linear expression is of the form $e : \vec{a}^T\vec{x}$ for vector $\vec{a} \in \mathbb{R}^n$. A linear inequality is of the form $\vec{a}^T\vec{x} \leq b$ and a linear assertion is a finite conjunction of linear inequalities $(\vec{a_1}^T\vec{x} \leq b_1 \ \wedge \ \cdots \ \wedge \ \vec{a_k}^T\vec{x} \leq b_k)$ conveniently written in matrix form as $A\vec{x} \leq \vec{b}$.

Definition 2.3.2 (Linear Transition Systems)**.** A linear transition system (LTS) is a transition system with the following restrictions:

(1) The initial conditions and transition guards are all linear assertions over X

(2) The update function for each transition is an affine function: $g_i(\vec{x}) : U_i\vec{x} + \vec{v_i}$.

Throughout this chapter, we will tackle linear transition systems. An error specification is written as $\langle \ell, \psi \rangle$ for a location ℓ and a linear assertion ψ. The goal is to prove that no reachable state for location ℓ satisfies ψ. I.e, all reachable states $\vec{x}$ at location ℓ satisfy $\vec{x} \notin [\![\psi]\!]$. To prove a given specification, we use an **inductive invariant**.

Definition 2.3.3 (Inductive Invariant Map)**.** An inductive invariant map $\eta : \mathcal{L} \to \Pi[X]$ maps each location $\ell \in \mathcal{L}$ to an assertion $\eta(\ell)$ such that the following conditions hold:

- **Initial Condition:** At the initial location ℓ_0, the entailment $\Theta \models \eta(\ell_0)$ holds.
- **Consecution Condition:** For each transition $\tau : \langle \ell_1, \ell_2, \psi_i, g_i \rangle$, the following consecution condition holds:

$$\eta(\ell_1) \ \wedge \ \psi_i \ \wedge \ \vec{x}' = g_i(\vec{x}) \ \models \ \eta(\ell_2)[\vec{x}'] \, .$$

  The condition states that starting from any state $\vec{x} \in [\![\eta(\ell_1)]\!]$, a single step of the transition τ, if enabled, yields a state $\vec{x}' \in [\![\eta(\ell_2)]\!]$.

Let η be an inductive assertion map and $\langle \ell, \psi \rangle$ be an error specification.

Theorem 2.3.1. *If the conjunction $\eta(\ell) \wedge \psi$ is unsatisfiable, then for every reachable state $(\ell, \vec{x})$, it follows that $\vec{x} \notin [\![\psi]\!]$.*

Proof. The proof first establishes that for any reachable state $\langle \ell, \vec{x} \rangle$ of the system, we have that $\vec{x} \in [\![\eta(\ell)]\!]$. In other words, the inductive invariants characterize all reachable states. Therefore, if $\eta(\ell) \wedge \psi$ is unsatisfiable, then no state can be reachable and satisfy ψ. □

The problem therefore consists of finding inductive assertion maps that can prove a given error specification.

2.3.1 Abstract Interpretation

Abstract interpretation provides a framework for systematically computing inductive assertions using a pre-specified lattice of assertions called an **abstract domain** [CC77, CC92]. The key insight lies in characterizing inductive assertion maps as post-fixed points of a monotone operator over sets of states.

In this section, we briefly sketch the basics of abstract interpretation, and the Kleene iteration using widening to compute post-fixed points.

The **concrete domain** Σ is a lattice whose elements are first order assertions over X, ordered by entailment $\models$. The logical disjunction $\vee$ is the join operator and conjunction $\wedge$ is the meet operator in this lattice. The bottom element is *false* and the top element is *true*. We define the post condition operation over sets of states and a transition τ.

Definition 2.3.4 (Post-Condition)**.** Given a set $\psi \in \Sigma$, its post condition with respect to a transition $\tau : \langle \ell_1, \ell_2, \varphi, g \rangle$ is the set of all states reachable from some state in $[\![\psi]\!]$ in one step by executing the transition τ:

$$post(\psi, \tau) : \ (\exists\ \vec{y})\ \psi(\vec{y}) \ \wedge\ \varphi(\vec{y}) \ \wedge\ \vec{x} = g(\vec{y})\,.$$

We consider assertion maps $\eta : \mathcal{L} \to \Sigma$ and let $\mathcal{N}$ be the set of all such maps. We lift the $\models$ operator from assertions to maps: $\eta_1 \models \eta_2$ iff for all $\ell \in \mathcal{L}$, $\eta_1(\ell) \models \eta_2(\ell)$. Thus, $\mathcal{N}$ forms a lattice

with the lifted $\models$ as the inclusion. Next, we define a monotone operator $\mathcal{F} : \mathcal{N} \to \mathcal{N}$ as

$$\mathcal{F}(\eta)(\ell) : \begin{cases} \Theta \ \vee \ \bigvee_{\tau:\langle m,\ell,\varphi,g\rangle} post(\eta(m),\tau) & \ell = \ell_0 \\ \bigvee_{\tau:\langle m,\ell,\varphi,g\rangle} post(\eta(m),\tau) & \text{otherwise} \end{cases}$$

An assertion map η is a post fixed point of $\mathcal{F}$ iff

$$\mathcal{F}(\eta) \models \eta\,.$$

Theorem 2.3.2. *An assertion map is inductive if and only if it is a post fixed point of $\mathcal{F}$.*

Proof. The proof is available in most textbooks on static analysis [NNH99]. A proof of this statement using the same notation as this section is available in the PhD thesis of Sankaranarayanan [San05] (Lemma 3.2). □

To compute a post-fixed point, we start with the bottom element of $\mathcal{N}$, an assertion map $\eta_\perp$ such that $\eta_\perp(\ell) = \mathit{false}$ for all $\ell \in \mathcal{L}$. We define the Kleene iteration as the sequence obtained by iterating $\mathcal{F}$ over $\eta_\perp$.

$$\eta^{(i)} : \begin{cases} \mathcal{F}(\eta_{i-1}) & i \geq 1 \\ \eta_\perp & i = 0 \end{cases}.$$

The process is stopped whenever $\eta^{(i+1)} \models \eta^{(i)}$, in which case, we can show that $\eta^{(i+1)} \equiv \eta^{(i)}$ is the **least fixed point**. However, the iteration may go on forever even for simple programs. To make matters worse, each step potentially yields larger and more complex formulas, making the computation of *post*, $\vee$ and $\models$ prohibitively expensive.

To counter this, abstract interpretation defines an abstract domain which is a lattice $\langle A, \sqsubseteq, \sqcup, \sqcap, \perp, \top\rangle$ with inclusion $\sqsubseteq$, join operator $\sqcup$, meet operator $\sqcap$, a bottom element $\perp$ and top element $\top$ wherein each element $a \in A$ is linked to the concrete domain through the concretization function $\gamma(a) \in \Sigma$. Likewise, each assertion $\psi \in \Sigma$ is linked to A through the abstraction function $\alpha(\gamma) \in A$. Together, the pair α, γ form a **Galois** connection:

$$(\forall a \in A,\ \varphi \in \Sigma)\ a \sqsubseteq \alpha(\varphi)\ \textit{iff}\ \gamma(a) \models \varphi\,.$$

The abstract post condition operation is defined as $\widehat{post}(a, \tau)$ with a **soundness condition**:

$$(\forall a \in A)\ post(\gamma(a), \tau) \models \gamma(\widehat{post}(a, \tau))\,.$$

Once again, we lift the domain A to a lattice over maps $N : \mathcal{L} \rightarrow A$. The abstract operator $\mathcal{G}$ is now defined analogous to the concrete operator $\mathcal{F}$.

$$\mathcal{G}(\widehat{\eta})(\ell) : \begin{cases} \alpha(\Theta) \sqcup \bigsqcup_{\tau:\langle m,\ell,\varphi,g\rangle} \widehat{post}(\widehat{\eta}(m), \tau) & \ell = \ell_0 \\ \bigsqcup_{\tau:\langle m,\ell,\varphi,g\rangle} \widehat{post}(\widehat{\eta}(m), \tau) & \text{otherwise} \end{cases}$$

A map $\widehat{\eta}$ is an abstract fixed point iff $\mathcal{G}(\widehat{\eta}) \sqsubseteq \widehat{\eta}$. The following theorem summarizes the core soundness property of abstract interpretation.

Theorem 2.3.3. *$\widehat{\eta}$ is an abstract post fixed point iff $\gamma \circ \widehat{\eta}$ is an inductive assertion map.*

The proof is available from most expositions of abstract interpretation [NNH99, CC77]. For a proof using the notation introduced in this section, we refer the reader to Theorem 3.1 of Sankaranarayanan's PhD thesis [San05].

Once again, the abstract Kleene iteration can be applied to compute a post fixed point in the abstract domain.

$$\widehat{\eta}^{(i)} : \begin{cases} \mathcal{G}(\widehat{\eta}_{i-1}) & i \geq 1 \\ \widehat{\eta}_{\perp} & i = 0 \end{cases}.$$

If the lattice A has the **ascending chain condition** property, then the process is guaranteed to converge, yielding an inductive assertion map. Otherwise, the process can still continue for ever. In this case, we use a widening operator to guarantee convergence. Formally, the widening operator $\nabla : A \times A \rightarrow A$ has the following properties:

(1) $a \sqcup b \sqsubseteq a \nabla b$ for all $a, b \in A$.

(2) For any non-decreasing sequence

$$a_0 \sqsubseteq a_1 \sqsubseteq \cdots$$

the corresponding widened sequence

$$b_0 : a_0, b_1 : \ b_0 \nabla a_1, \ \cdots \ b_{i+1} : \ b_i \nabla a_{i+1} \ \cdots$$

always converges in finitely many steps to yield $b_{i+1} \sqsubseteq b_i$.

In practice, widening causes an unacceptable loss in precision that is improved using a narrowing iteration. A narrowing operator Δ is used to terminate a descending sequence of lattice elements:

$$b_0 \sqsupseteq b_1 \sqsupseteq b_2 \cdots .$$

It satisfies the key property that if $a \sqsupseteq b$ then $a \sqsupseteq (a \Delta b) \sqsupseteq b$, and furthermore, the narrowed descending iteration

$$c_0 : \ b_0, \ c_1 : (c_0 \Delta b_1), \ c_2 : \ (c_1 \Delta b_2), \cdots, c_{j+1} : \ (c_j \Delta b_{j+1}) \cdots ,$$

terminates in finitely many steps.

2.3.2 Template Domains

The rest of this chapter will focus on the abstract domain of template polyhedra [SSM05]. Let $\mathcal{S} : \langle \mathcal{L}, X, \mathcal{T}, \ell_0, \Theta \rangle$ be a linear transition system. Let $\vec{x}$ represent the system variables in X as a vector and $n = |X|$.

A template associates each location $\ell \in \mathcal{L}$ with a $m_\ell \times n$ matrix T_ℓ. We drop the subscript ℓ from the template matrix if the location ℓ is clear from the context. A $m \times n$ template T defines a lattice $\mathcal{A}(T)$:

$$\mathcal{A}(T) : \{\vec{c} \in \mathbb{R}_+^m\}, \text{ wherein, } \gamma(\vec{c}) : \ T\vec{x} \leq \vec{c}.$$

In other words, each element of the template abstract domain is a possible valuation $\vec{c}$ to the RHS of inequalities $T\vec{x} \leq \vec{c}$. Note that the entries in $\vec{c}$ can include $\pm\infty$. Naturally, we define the linear inequality $e \leq \infty$ to be synonymous with *true* and $e \leq -\infty$ is synonymous with *false*.

Given an assertion φ over $\vec{x}$, its abstraction $\vec{c} : \alpha(\varphi)$ is computed as a vector whose i^{th} entry $\vec{c_i}$ is the solution to the optimization problem:

$$\vec{c_i} : \ \max \ T_i\vec{x} \ \mathsf{s.t.} \ \varphi(\vec{x}) .$$

Since the abstraction is often computed for linear assertions φ, this is a linear programming (LP) problem.

For each template element, its **canonical representative** $\mathsf{can}(\vec{c})$ is defined as the instantiation $\vec{d}$, whose i^{th} entry $\vec{d_i}$ is the solution to the following LP:

$$\vec{d_i} : \ \max \ T_i\vec{x} \ \mathsf{s.t.} \ T\vec{x} \leq \vec{c}.$$

Note that the solution to an unbounded problem is taken to be $+\infty$ and an infeasible problem to be $-\infty$. Note that the template polyhedron defined by $T\vec{x} \leq \vec{c}$ is identical to the polyhedron $T\vec{x} \leq \mathsf{can}(\vec{c})$. A template element $\vec{c}$ is **canonical** in $\mathcal{A}(T)$ if and only $\vec{c} = \mathsf{can}(\vec{c})$.

The inclusion operator $\sqsubseteq$ in $\mathcal{A}(T)$ is defined as

$$\vec{c_1} \sqsubseteq \vec{c_2} \text{ iff } \mathsf{can}(\vec{c_1}) \leq \mathsf{can}(\vec{c_2}) \,,$$

wherein $\leq$ operation over vectors compares elements entrywise. The join operator $\vec{c_1} \sqcup \vec{c_2}$ is simply the entrywise maximum $\max(\vec{c_1}, \vec{c_2})$. Likewise, the meet operator is the canonical entrywise minimum.

Let T_ℓ be the template associated with location ℓ and T_m with location m. The abstract post with respect to a transition $\left\langle \ell, m, \varphi : A\vec{x} \leq \vec{b}, g : U\vec{x} + \vec{v} \right\rangle$ is an operator $\widehat{post} : \mathcal{A}(T_\ell) \times \mathcal{T} \ \rightarrow \ \mathcal{A}(T_m)$. Given $\vec{c} \in \mathcal{A}(T_\ell)$, the result $\vec{d} : \widehat{post}(\vec{c}, \tau)$ is a vector wherein $\vec{d_i}$ is given as the solution to the following LP:

$$\vec{d_i} : \left(\begin{array}{lll} \max & T_{m,i}\vec{x} & \\ \mathsf{s.t.} & T_\ell\vec{y} & \leq \vec{c} \\ & A\vec{y} & \leq \vec{b} \\ & \vec{x} & = U\vec{y} + \vec{v} \end{array} \right)$$

Widening and narrowing operators for the template domain are defined by extensions of the standard interval widening operator [SSM05].

The template domain is a convenient numerical abstract domain that uses linear programming solvers as a primitive for implementing the domain operations. However, a common critique of the template approach is that it requires users to specify the template T. In practice, users default

to popular choices such as **intervals**, **octagons** and **pentagons** which avoid repeated calls to LP solvers by using special properties of the constraints in these templates. We proceed by assuming that an initial template has been specified for each location using one of the schemes outlined above. Our approach can change this template as part of the solution scheme.

2.4 Bilinear Constraints and Policy Iteration

In this section, we consider the data flow equations for template abstract domain, connecting them to a class of nonconvex optimization problems called **bilinear optimization problem** (BOP). We present the **policy iteration** approach, proposed by Gaubert et al. as a technique for solving such bilinear inequalities that alternates between solving linear programs [GGTZ07]. Once again we fix a linear transition system $\mathcal{S}$ and assume for simplicity that each location ℓ is labeled with the same $m \times n$ matrix T. The approach can be easily extended to the case where the template matrices differ between locations.

We will make use of Farkas' lemma, a standard result in linear programming. Let $\varphi : \ A\vec{x} \leq \vec{b}$ be a linear assertion with $m \times n$ matrix A and $m \times 1$ vector $\vec{b}$, $\psi : \ \vec{c}^T\vec{x} \leq d$ be a given linear inequality.

Theorem 2.4.1 (Farkas Lemma)**.** *If φ is satisfiable, then $\varphi \models \psi$ iff there exists nonnegative multipliers $\vec{\lambda} \in \mathbb{R}^m$ such that*

$$A^T\vec{\lambda} = \vec{c} \ \wedge \ \vec{b}^T\vec{\lambda} \leq d \ \wedge \ \vec{\lambda} \geq 0\,. \tag{2.1}$$

Furthermore, φ is unsatisfiable if and only if there exists multipliers $\vec{\lambda} \in \mathbb{R}^m$ such that

$$A^T\vec{\lambda} = \vec{0} \ \wedge \ \vec{b}^T\vec{\lambda} \leq -1 \ \wedge \ \vec{\lambda} \geq 0\,.$$

The constraints can be seen as encoding the entailment $\varphi \models \vec{0}^T\vec{x} \leq -1$.

A proof can be found in most textbooks that deal with linear optimization [Chv83]. Given a system of constraints $\varphi : A\vec{x} \leq \vec{b}$, the form in Eq. (2.1) is often referred to as the dual. Furthermore, the multipliers $\vec{\lambda}$ are often referred to as as the dual variables or dual multipliers.

Note that Farkas lemma handles the entailment of a single linear inequality. However, for a polyhedron $C\vec{x} \leq \vec{d}$, we may encode the entailment $A\vec{x} \leq \vec{b} \models C\vec{x} \leq \vec{d}$ as a series of single inequality entailments: $A\vec{x} \leq \vec{b} \models C_j\vec{x} \leq \vec{d_j}$ for each row j of $C, \vec{d}$. The resulting constraints can be collectively written as:

$$A^T\Lambda = C,\ \Lambda^T\vec{b} \leq \vec{d},\ \Lambda \geq 0\,.$$

All equalities and inequalities between matrices are interpreted entrywise. Here Λ is a matrix with as many rows as A and as many columns as the number of rows in C. The j^{th} column of Λ contains the multipliers corresponding to the inequality $C_j\vec{x} \leq \vec{d_j}$. This notation will be used throughout the rest of the chapter.

Using Farkas' lemma, we may now derive a system of constraints corresponding to the **data flow equations** for the template domain. Let T be a $m \times n$ template matrix. We associate each location ℓ with an unknown vector $\vec{c}(\ell) \in \mathcal{A}(T)$ such that the assertion map $\eta(\ell) :\ T\vec{x} \leq \vec{c}(\ell)$ is inductive.

We wish to encode the constraints for initiation:

$$\Theta \models T\vec{x} \leq \vec{c}(\ell_0)\,, \tag{2.2}$$

and for each transition $\tau : \langle \ell, m, \varphi, g\rangle$, we wish to model consecution:

$$T\vec{x} \leq \vec{c}(\ell)\ \wedge\ \varphi\ \wedge\ \vec{x}' = g(\vec{x})\ \models\ T\vec{x}' \leq \vec{c}(m)\,. \tag{2.3}$$

Initiation: Let $\Theta : A_0\vec{x} \leq \vec{b_0}$ be the assertion for the initial condition. Using Farkas' lemma for the entailment in Eq. (2.2), we obtain the condition:

$$A_0^T\Lambda_0 = T\ \wedge\ \Lambda_0^T\vec{b_0} \leq \vec{c}(\ell_0)\ \wedge\ \Lambda_0 \geq 0\,. \tag{2.4}$$

Here Λ_0 is a $k \times m$ matrix wherein k is the number of rows in A_0 and m is the number of rows in T. We write $\Lambda_0 \geq 0$ to indicate that all entries in Λ_0 are non-negative.

Consecution: Let τ be a transition with guard $A_\tau\vec{x} \leq \vec{b_\tau}$ and update $g(\vec{x}) : U_\tau\vec{x} + \vec{v_\tau}$. The consecution condition in Eq. (2.3) can be rewritten through substitution of $\vec{x}'$ and arranged as

follows:

$$\begin{array}{r|rcl} \Lambda_\tau \rightarrow & T\vec{x} & \leq & \vec{c}(\ell) \\ \Gamma_\tau \rightarrow & A_\tau\vec{x} & \leq & \vec{b}_\tau \\ \hline \models & TU_\tau\vec{x} & \leq & \vec{c}(m) - T\vec{v}_\tau \end{array}$$

The notation above shows the constraints and the associated dual multipliers with each block of constraints. Furthermore, we have substituted $\vec{x}' = U_\tau\vec{x} + \vec{v}_\tau$. This is dualized using Farkas' lemma to yield the following constraints:

$$\begin{array}{rcl} T^T\Lambda_\tau + A_\tau^T\Gamma_\tau & = & TU_\tau \\ \Lambda_\tau^T\vec{c}(\ell) + \Gamma_\tau^T\vec{b}_\tau & \leq & \vec{c}(m) - T\vec{v}_\tau \\ \Lambda_\tau, \Gamma_\tau & \geq & 0 \end{array} \tag{2.5}$$

Note that Eq. (2.4) for the initiation yields a system of linear constraints involving $\vec{c}(\ell_0)$ and unknown multipliers in Λ_0. However, the consecution constraints in Eq. (2.5) for each transition τ involve the product $\Lambda_\tau^T\vec{c}(\ell)$ both of which are unknown. This makes the constraints for consecution fall into a special class called **bilinear constraints**. I.e., for a fixed Λ_τ these constraints are linear in the remaining variables $\vec{c}(\ell), \Gamma_\tau$. Similarly, for fixed values of $\vec{c}(\ell)$, these constraints are linear in the variables $\Lambda_\tau, \Gamma_\tau$. Figure 2.3 summarizes the constraints obtained at a glance. Note that the multipliers Λ_τ are called **bilinear multipliers** since they are multiplied with the unknowns $\vec{c}(\ell)$ to form the nonlinear terms in the constraints. On the other hand, note that Λ_0, Γ_τ variables are not multiplied with other unknowns.

Connection with Min-Policies: The original "min-policy" approach of Costan et al. [CGG$^+$05] considers data flow equations of the form:

$$\vec{c} \geq \min(\vec{a}_{i,1}^T\vec{c}, \ldots, \vec{a}_{i,k}^T\vec{c}), \;\; i = 1, \ldots, M, \;\; k = 1, \ldots, N\,. \tag{2.6}$$

We will demonstrate that the equations shown in Figure 2.3 can be equivalently expressed in this form. For simplicity, we consider the case for a single location ℓ with template T and unknown template RHS variables $\vec{c}$. All transitions are assumed to be self-loops around this location. From

Figure 2.3: Bilinear system of constraints at a glance. The constraints are generalized to allow for possibly different templates T_ℓ at each location.

TemplateVars :	$\vec{c}(\ell),\ \ell \in \mathcal{L}$	
BilinearMults :	$\Lambda_\tau,\ \tau \in \mathcal{T}$	
LinearMults :	$\Lambda_0, \Gamma_\tau,\ \tau \in \mathcal{T}$	
Constraints :	$A_0^T \Lambda_0 = T_{\ell_0}$	(* Initiation *)
	$\Lambda_0^T \vec{b}_0 \leq \vec{c}(\ell_0)$	
	$T_l^T \Lambda_\tau + A_\tau^T \Gamma_\tau \ = \ T_m U_\tau$	(* Consecution $\tau : \langle l, m, \varphi, g \rangle$ *)
	$\Lambda_\tau^T \vec{c}(\ell) + \Gamma_\tau^T \vec{b}_\tau \ \leq \ \vec{c}(m) - T_m \vec{v}_\tau$	
	$\Lambda_0, \Lambda_\tau, \Gamma_\tau \geq 0$	(* Nonnegative multipliers *)

eq. (2.5), a given solution $\vec{c}$ satisfies the consecution for transition τ iff there exist $\Lambda_\tau, \Gamma_\tau$ such that

$$\vec{c} \ \geq \ \Lambda_\tau^T \vec{c} + \Gamma_\tau^T \vec{b}_\tau + T\vec{v}_\tau \tag{2.7}$$

$$T^T \Lambda_\tau + A_\tau^T \Gamma_\tau \ = \ T U_\tau \tag{2.8}$$

$$\Lambda_\tau, \Gamma_\tau \ \geq 0 \tag{2.9}$$

Let us define a polyhedron $P(\Lambda_\tau, \Gamma_\tau)$ defined by collecting the constraints in lines (2.8), (2.9) above. We may rewrite the constraints equivalently as:

$$\vec{c} \ \geq \ \min_{(\Lambda_\tau, \Gamma_\tau) \in P} \left(\Lambda_\tau^T \vec{c} + \Gamma_\tau^T \vec{b}_\tau + T\vec{v}_\tau \right) \tag{2.10}$$

Note that P is a polyhedron. Let us assume that it is defined by N vertices:

$$(\Lambda_1, \Gamma_1), \ \ldots, \ (\Lambda_N, \Gamma_N) \ .$$

The min in eq. (2.10) can be equivalently written as a minimization over the finite set of vertices of P:

$$\vec{c} \ \geq \ \min_{j=1}^{N} \left(\Lambda_j^T \vec{c} + \Gamma_j^T \vec{b}_\tau + T\vec{v}_\tau \right) \tag{2.11}$$

We note that this form arises from the specific structure of the data flow equations for the template abstract domain. In particular, not all bilinear constraints satisfy this property.

2.4.1 Policy Iteration

We now describe policy iteration as an alternation between solving for unknown $\vec{c}(\ell)$ for each $\ell \in \mathcal{L}$ and solving for the unknown bilinear multipliers Λ_τ. Policy iteration starts from a known sound solution $\vec{c}^0(\ell)$ and successively improves the solution to obtain better solutions (smaller in the lattice) until no further improvements can be obtained. The initial solution may be obtained by using Kleene iteration with widening. For simplicity, we will assume that $\vec{c}^0(\ell) \neq \bot$, for each $\ell \in \mathcal{L}$. If this were the case, then the location ℓ is unreachable, and can be removed from the system.

The overall scheme alternates between (I) **solving for the unknown multipliers** $\Lambda_\tau, \Gamma_\tau, \Lambda_0$ given a fixed value of $\vec{c}$, and (II) **solving for the unknown template RHS** $\vec{c}(\ell)$ given $\Lambda_\tau, \Gamma_\tau$ and Λ_0. Since Γ_τ and Λ_0 are not involved in any bilinear term, we do not fix them to specific values when solving for $\vec{c}(\ell)$.

Solving for Multipliers: Given the values for the current solution $\vec{c}^{(i)}(\ell)$ at each location, we simply plug in these values and solve the system in Figure 2.3.

Lemma 2.4.2. *The constraints shown in Fig. 2.3 become linear if we replace $\vec{c}(\ell)$ at each location by fixed (constant) values.*

Proof. Proof is by inspection. We run through each inequality and note that the constraints are linear in the variables $\vec{c}^{(i)}(\ell)$ and the multipliers Λ_0, Λ_τ, and Γ_τ. □

The remaining constraints are linear over Λ_0, Λ_τ and Γ_τ for each transition τ, and can be thus solved using a LP solver. The following lemma guarantees that the constraints will always yield a feasible solution provided the values $\vec{c}^{(i)}$ are a valid post-fixed point.

Lemma 2.4.3. *If the solution $\vec{c}^{(i)}(\ell)$ for each $\ell \in \mathcal{L}$ is a post-fixed point, the constraints in the Fig. 2.3 are feasible for the remaining multipliers, when $\vec{c}(\ell)$ is replaced by $\vec{c}^{(i)}(\ell)$.*

Proof. Let us assume that the solutions $\vec{c}^{(i)}(\ell)$ yield a post-fixed point over the template polyhedra domain. We now wish to show that the constraints in Fig. 2.3 are feasible when we replace each $\vec{c}(\ell)$ by $\vec{c}^{(i)}(\ell)$.

The proof is obtained by combining two facts: (a) because $\vec{c}^{(i)}(\ell)$ form a post-fixed point, the initiation condition (Eq. (2.2)) and consection conditions for each transition $\tau \in \mathcal{T}$ (Eq. (2.3)) for inductive invariance hold. (b) Because these entailments hold, we can apply Farkas' lemma to each of them and derive the existence of multipliers for Eq. (2.4) and Eq. (2.5). However, the constraints in Figure 2.3 are just a conjunction of these constraints. Therefore, we conclude that there exist values of the multipliers that satisfy these constraints when we substitute $\vec{c}^{(i)}(\ell)$ for each $\vec{c}(\ell)$. □

Let $\Lambda_\tau^{(i)}$ be the resulting values of the bilinear multipliers returned by the LP solver when we replace $\vec{c} : \vec{c}^{(i)}$. These are also called policies [GGTZ07].

Solving for Template RHS: Next, let us assume that the variables Λ_τ for each transition are set to constants $\Lambda_\tau^{(i)}$.

Lemma 2.4.4. *If we set Λ_τ for each τ to constants $\Lambda_\tau^{(i)}$ for the constraints in Figure 2.3, the resulting problem is linear over $\vec{c}(\ell)$ for each $\ell \in \mathcal{L}$ and the linear multipliers Γ_τ, Λ_0.*

Proof. Proof is once again by inspection of each constraint in Figure 2.3. □

Once we set Λ_τ to specific values, the resulting system is once again a linear program. Let us call this problem $\mathcal{C}_i$.

Lemma 2.4.5. *The LP $\mathcal{C}_i$ is always feasible.*

Proof. To see this, we note that $\vec{c}(\ell) = \vec{c}^{(i)}$ is already a solution to this LP due to how the values of $\Lambda_\tau^{(i)}$ were obtained in the first place. We call the resulting values $\vec{c}^{(i+1)}(\ell)$.

□

The overall policy iteration scheme alternates between solving for $\vec{c}(\ell)$ and solving for Λ_τ variables. Gaubert et al. show that the number of policies needed is finite (but large), and thus the process is guaranteed to yield a stable solution such that $\vec{c}^{(i+1)}(\ell) = \vec{c}^{(i)}(\ell)$.

2.5 Policies with Template Update

In this section, we extend policy iteration process to achieve two goals simultaneously: (a) be goal-directed towards a specific property and (b) allow the template T at each location to be updated.

Let (ℓ, ψ) be a error specification at location ℓ that we wish to prove unreachable. Our goal is to compute an inductive assertion map η such that at location ℓ, the conjunction $\eta(\ell) \ \wedge \psi$ is unsatisfiable. Once again, we will first assume for the sake of exposition that the same template matrix T is used at each location.

Using Farkas' lemma, the invariant $T\vec{x} \leq \vec{c}(\ell)$ proves the unreachability of the error specification $\psi : P\vec{x} \leq \vec{q}$ iff there exist multipliers $\vec{\lambda}_s, \vec{\gamma}_s \geq 0$ s.t.

$$T^{\intercal}\vec{\lambda}_s + P^T\vec{\gamma}_s = \vec{0}, \ \underbrace{\vec{c}(\ell)^T\vec{\lambda}_s + \vec{q}^T\vec{\gamma}_s \leq -1}_{I}, \ \vec{\lambda}_s, \vec{\gamma}_s \geq 0\,. \tag{2.12}$$

However, if the invariant fails to prove the property, we will be unable to find suitable multipliers $\vec{\lambda}_s, \vec{\gamma}_s \ \geq 0$. Since, our procedure will involve intermediate solutions that do not satisfy the property, we will consider the following optimization-based formulation by moving the inequality labeled "I" in (2.12) to the objective, as follows:

$$\begin{array}{lrcl} \min & \vec{c}(\ell)^T\vec{\lambda}_s + \vec{q}^T\vec{\gamma}_s & & \\ \mathsf{s.t.} & T^T\vec{\lambda}_s + P^T\vec{\gamma}_s & = & \vec{0} \\ & \vec{1}^T\vec{\lambda}_s & = & 1 \quad \text{(* normalization constraint *)} \\ & \vec{\lambda}_s, \vec{\gamma}_s & \geq & 0 \end{array} \tag{2.13}$$

Note that we have added a **normalization constraint** requiring that the sum of the multipliers $\vec{\lambda}_s$ equal 1. Without such a constraint, the problem always has a trivial solution 0 by setting all the multipliers $(\vec{\lambda}_s, \vec{\gamma}_s)$ to 0, which is undesirable for the policy iteration scheme to be discussed subsequently.

Lemma 2.5.1. *Suppose $T_i = -P_j$ for row i of matrix T, row j of matrix P, and $\vec{c}(\ell)_i < \infty$ then the optimization problem in Eq.* (2.13) *is feasible.*

Furthermore, its objective value is strictly negative iff $T\vec{x} \leq \vec{c}(\ell)$ *proves the specification* $(\ell, \psi : P\vec{x} \leq \vec{q})$.

Proof. Given that $T_i = -P_j$, we then choose $\vec{\lambda}_s(i) = 1$ and the rest of entries to zero. Likewise, $\vec{\gamma}_s(j) = 1$ and the remaining entries of $\vec{\gamma}_s$ are set to 0. We can now verify that this will satisfy the constraints, thus providing a feasible solution.

Note that if we find a solution $(\vec{\lambda}_s, \vec{\gamma}_s)$ such that the objective value is $\epsilon < 0$, then $(\frac{\vec{\lambda}_s}{|\epsilon|}, \frac{\vec{\gamma}_s}{|\epsilon|})$ satisfy the constraints in Eq. (2.12). The rest follows from Farkas' lemma. □

Thus, we will use the optimization formulation as an objective function that measures how "far away" the current solution at ℓ is from proving the property of interest.

2.5.1 Updating Templates

Next, we allow the template T to change at each step to a new template $T^{(i+1)} : \ T^{(i)} + \Delta$, starting with the initial template $T^{(0)} = T$, wherein Δ is the unknown change in the template. As a result, our analysis explores a series of templates:

$$T^{(0)},\ T^{(1)}, \ldots, T^{(N)}\,.$$

In doing so, we update the constraints to introduce an unknown change Δ. However, allowing arbitrary changes to the template will not work since choosing $\Delta = -T^{(i)}$ immediately makes the template trivial, and not useful for our purposes. Therefore, we specify upper and lower limits to the change in the template. These limits can be set using different strategies that we will explore in the experimental evaluation section. Let L be the lower limit and U be the upper limit so that $L \leq T^{(i)} + \Delta \leq U$. As a technical condition, we require $T^{(i)} \in [L, U]$, i.e., the option to keep the current template $T^{(i)}$ unchanged is allowed at each step.

Figure 2.4 shows the bilinear optimization problem

$$\mathcal{B}\left((\vec{c}(\ell), \Delta_\ell), (\Lambda_\tau, \vec{\lambda}_s)\right)\,,$$

obtained when the change in the template variables is also considered. Here note that ℓ ranges over all location, τ over all transitions and finally, $\vec{\lambda}_s$ pertains to the property to be checked which is assumed to be relevant to a single location in the program. We note that the variables involved in the bilinear terms are once again separated into two sets, represented in different colors for convenience.

2.5.2 Template Updates and Policy Iteration

We now update the policy iteration process to consider the change in templates, as shown in Fig. 2.4. Let $\vec{c}^{(0)}$ be an initial value such that $T_\ell^{(0)}\vec{x} \leq \vec{c}^{(0)}(\ell)$ is inductive. The initial template $T_\ell^{(0)}$ is specified by the user, and furthermore, the initial inductive invariant is assumed to be computed by abstract interpretation. The initial update $\Delta_\ell^{(0)} = 0$ for each location ℓ.

Note: For convenience, we will assume that $(\vec{c}^{(0}(\ell))_i \neq \pm\infty$. Indeed, if any entry of $\vec{c}_\ell^{(i)}$ is $-\infty$, then the location ℓ is deemed unreachable and removed from the transition system. Also, if any entry of $\vec{c}_\ell^{(i)}$ is $+\infty$, we will simply remove the corresponding template row from our analysis.

Multiplier Update: At each iteration i, the multiplier update uses $\vec{c}^{(i)}, \Delta^{(i)}$ to obtain values of $\Lambda_\tau^{(i)}, \vec{\lambda}_s^{(i)}$. Formally, we consider the problem

$$\mathcal{M}_i : \ \mathcal{B}\left((\vec{c}^{(i)}(\ell), \Delta_\ell^{(i)}), (\Lambda_\tau, \vec{\lambda}_s)\right)$$

Lemma 2.5.2. *The following are true:*

(1) $\mathcal{M}_i$ is a linear program over unknown multipliers $\Lambda_\tau, \vec{\lambda}_s, \Gamma_\tau, \vec{\gamma}_s, \Lambda_0$.

(2) It is feasible iff the map $\eta^{(i)}$ formed by the assertions $(T_\ell^{(i)} + \Delta_\ell^{(i)})\vec{x} \leq \vec{c}^{(i)}(\ell)$ for $\ell \in \mathcal{L}$, is an inductive assertion map.

(3) The value of the objective function cannot increase, i.e., for $i > 1$,

$$\vec{c}^{(i)}(\ell)^T\vec{\lambda}_s^{(i)} + \vec{q}^T\vec{\gamma}_s^{(i)} \leq \vec{c}^{(i)}(\ell)^T\vec{\lambda}_s^{(i-1)} + \vec{q}^T\vec{\gamma}_s^{(i-1)}\,.$$

(4) The value of the objective is negative iff $\eta^{(i)}$ proves the specification (ℓ, ψ).

Proof. **(1)** First, to see that $\mathcal{M}_i$ is a linear program, we inspect the constraints in Figure 2.4. Each constraint is linear in the unknowns $(\Lambda_\tau, \vec{\lambda}_s)$.

(2) The proof is identical to that of Lemma 2.4.3. Suppose, the map $\eta : \ell \rightarrow (T_\ell^{(i)} + \Delta_\ell^{(i)})\vec{x} \leq \vec{c}^{(i)}(\ell)$ foreach $\ell \in \mathcal{L}$, forms an inductive assertion map. We note that η satisfies the conditions for initiation (2.4) and consecution (2.5) with the template $T_\ell^{(i)}$ replaced by $T_\ell^{(i+1)} : T_\ell^{(i)} + \Delta_\ell^{(i)}$. Dualizing the initiations and consecution conditions using Farkas lemma yields the constraints in Fig. 2.4. As a result, we note that when we substitute values $\Delta_\ell^{(i)}$ for each Δ_ℓ and $\vec{c}^{(i)}(\ell)$ for each $\vec{c}(\ell)$, the multipliers have a feasible solution provided by Farkas' lemma.

(3) We note that $\vec{\lambda}_s^{(i-1)}, \vec{\gamma}_s^{(i-1)}$ form (part of a) feasible solution for $\mathcal{M}_i$ and furthermore, $\vec{\lambda}_s^{(i)}$ and $\vec{\gamma}_s^{(i)}$ form part of the optimal solution for the same. Since, we are minimizing the objective, the optimal objective must be less than or equal to any feasible solution.

(4) The value of the objective is negative iff $\eta^{(i)}$ proves the specification (ℓ, ψ). □

The result of multiplier update yields values for the variables $(\Lambda_\tau, \vec{\lambda}_s) : (\Lambda_\tau^{(i)}, \vec{\lambda}_s^{(i)})$.

Template Update: Given the current values $(\Lambda_\tau^{(i)}, \vec{\lambda}_s^{(i)})$ for the multipliers, we derive new values $\vec{c}^{(i+1)}(\ell), \Delta_\ell^{(i+1)}$ for the template variables by solving the problem

$$\mathcal{C}_{i+1} : \mathcal{B}\left((\vec{c}(\ell), \Delta_\ell), (\Lambda_\tau^{(i)}, \vec{\lambda}_s^{(i)})\right) .$$

Lemma 2.5.3. *The following facts hold:*

(1) $\mathcal{C}_{i+1}$ is a linear program over the unknown template variables $\vec{c}(\ell), \Delta_\ell$ and unknown linear multipliers $\Gamma_\tau, \vec{\gamma}_s, \Lambda_0$.

(2) It is always feasible provided $T^{(i)} \in [L_\ell, U_\ell]$ at each location at each iteration.

(3) The assertion map $\eta^{(i+1)}$ formed by the solution

$$(T_\ell^{(i)} + \Delta_\ell^{(i+1)})\vec{x} \leq \vec{c}^{(i+1)}(\ell) \text{ for } \ell \in \mathcal{L} ,$$

is inductive.

(4) The value of the objective function cannot increase, i.e., for $i \geq 0$,

$$\vec{c}^{(i+1)}(\ell)^T \vec{\lambda}_s^{(i)} + \vec{q}^T \vec{\gamma}_s^{(i+1)} \leq \vec{c}^{(i)}(\ell)^T \vec{\lambda}_s^{(i)} + \vec{q}^T \vec{\gamma}_s^{(i)} .$$

(5) The value of the objective function $\vec{c}^{(i+1)}(\ell)^T \vec{\lambda}_s^{(i)} + \vec{q}^T \vec{\gamma}_s^{(i+1)}$ *is negative iff the* $\eta^{(i+1)}$ *proves the property.*

Proof. Proof is entirely analogous to Lemma 2.5.2.

(1) We note that $\mathcal{C}_i$ is a linear program since each constraint in Figure 2.4 is linear in the unknowns $(\vec{c}(\ell), \Delta_\ell)$.

(2) We note that setting $\Delta_\ell = 0, \vec{c}(\ell) = \vec{c}^{(i)}(\ell)$ yields a feasible solution for $\mathcal{C}_i$.

(3) We note that the assertion map $\eta^{(i+1)}$ formed by the solution

$$(T_\ell^{(i)} + \Delta_\ell^{(i+1)})\vec{x} \leq \vec{c}^{(i+1)}(\ell) \text{ for } \ell \in \mathcal{L}\,,$$

satisfies the dual of the initiation and consecution constraints due to the existence of the multipliers, and thus by Farkas lemma satisfies the conditions themselves.

(4) We note that $\vec{c}^{(i)}(\ell), \Delta_\ell = 0$ is already feasible for $\mathcal{C}_i$. Therefore, the objective value achieved by the optimal solution must be less than or equal to that achieved by any feasible solution.

(5) This follows directly from Lemma 2.5.1.

□

The overall scheme alternates between updating the multipliers and the template variables, until no more changes can occur. We also observe that starting from a valid inductive invariant, the solutions obtained during the policy iteration continue to remain inductive or post-fixed points. However, they are post-fixed points over the lattice $\mathcal{A}(T_\ell^{(i)} + \Delta_\ell^{(i)},\ \ell \in \mathcal{L})$, which is different from the original lattice. As observed already in the motivating example (section 2.2), these invariants can be mutually incomparable. However, we show that at each step, the value of the objective function measuring progress towards proving the specification cannot increase.

2.5.3 Discussion

We now focus on issues such as convergence and the complexity of each step.

Convergence: In general, the known results about the convergence of alternating minimization schemes for bilinear optimization problems indicate that the process **seldom** converges to a global

optimal value [HM97]. Often, these iterations get "stuck" in a local **saddle point**, from which no further progress is possible. Nevertheless, our goal here is not to converge to a global optimum but to a **good enough** solution whose objective function value is strictly negative, thus proving the property of interest.

Example The following example taken from Adjé et al. [AGG14] demonstrates the fundamental difficulties of using policy iteration to calculate a "good" invariant.

var x **initially** 0 <= x <= 1
while (x <= 100)
    x := (1 - x)
end

The program can be written in the formalism of this chapter as a single location ℓ representing the head of the while loop and a single transition τ with guard $x \leq 100$ and update $x' = 1 - x$.

Since the program has a single variable, the "best" template possible using linear expressions is equivalent to the interval domain.

Policy iteration does not converge to the best solution $0 \leq x \wedge x \leq 1$, unless it is directly initialized to this invariant. For example, starting from the initial solution $0 \leq x \leq 100$, it converges instead to $-99 \leq x \leq 100$. ■.

By allowing template updates to the process, the problem is worsened in a sense. It is no longer clear that the process will necessarily converge (even if it converges to a saddle point) in finitely many steps. It is entirely possible that the value of the objective function remains unchanged but the process produces a new template $T_\ell^{(i)} + \Delta_\ell^{(i)}$ at each step. Depending on how the limits to the template change L_ℓ, U_ℓ are specified, this process may produce a fresh new template at each step.

Nevertheless, we note that the lack of convergence does not pose a serious hurdle to an application of template update to policy iteration. It is possible to iterate while each step provides at least $\epsilon > 0$ decrease in the value of the objective function, and stop otherwise.

Complexity: At each step, we solve a linear programming problem. For a transition system with n variables, $|\mathcal{L}|$ locations, $|\mathcal{T}|$ transitions, k template rows at each step, the size of each LP in terms of number of variables + constraints is $\mathcal{O}\left(|\mathcal{L}|kn + |\mathcal{T}|k^2\right)$. Although this is polynomial, the process can be prohibitively expensive for large programs. In our future work, we wish to exploit the block structure of these constraints in order to allow us to solve the LPs using standard approaches such as Benders or Danzig-Wolfe decomposition techniques [Chv83]

Collecting Invariants: Finally, we note that each step yields an invariant map $\eta^{(i)}$ that is not necessarily comparable to the invariant obtained in the next step $\eta^{(i+1)}$. However, we note that the finite conjunction

$$\eta^{(0)} \ \wedge \ \cdots \ \wedge \ \eta^{(N)},$$

over all the iterations of this process can be a stronger invariant than each of them. This is already demonstrated by the motivating example in Section 2.2.

Figure 2.4: Bilinear system of constraints with objective function and template update variables Δ_l. The current template after the i^{th} iteration at location $m \in \mathcal{L}$ is denoted $T_m^{(i)}$.

Vars :	$\vec{c}(\ell),\ \ell \in \mathcal{L}$	(* Template RHS *)
	$\Delta_\ell,\ \ell \in \mathcal{L}$	(* Template update *)
	$\Lambda_\tau,\ \tau \in \mathcal{T}$	(* Bilinear mult.*)
	$\vec{\lambda}_s$	(* Error Spec.*)
	$\Lambda_0, \Gamma_\tau,\ \tau \in \mathcal{T}$	(* Linear Mults. *)
	$\vec{\gamma}_s$	(* Error Spec.*)
min :	$\vec{c}(\ell)^T \vec{\lambda}_s + \vec{q}^T \vec{\gamma}_s$	
s.t.	$A_0^T \Lambda_0 = T_{\ell_0}^{(i)} + \Delta_{\ell_0}$	(* Initiation *)
	$\Lambda_0^T \vec{b}_0 \leq \vec{c}(\ell_0)$	
	$(T_\ell^{(i)} + \Delta_\ell)^T \Lambda_\tau + A_\tau^T \Gamma_\tau \ = \ (T_m^{(i)} + \Delta_m) U_\tau$	(* Consecution $\tau : \langle \ell, m, \varphi, g \rangle$ *)
	${\Lambda_\tau}^T \vec{c}(\ell) + \Gamma_\tau^T \vec{b}_\tau \ \leq \ \vec{c}(m) - (T_m^{(i)} + \Delta_m)\vec{v}_\tau$	
	$(T_\ell^{(i)} + \Delta_\ell)^T \vec{\lambda}_s + P^T \vec{\gamma}_s = 0$	(* Error spec. $\ell,\ \psi : P\vec{x} \leq \vec{q}$ *)
	$\mathbf{1}^T \vec{\lambda}_s = 1$	
	$\Lambda_0, \Lambda_\tau, \vec{\lambda}_s, \Gamma_\tau, \vec{\gamma}_s \geq 0$	(* Nonnegative multipliers *)
	$L_\ell \leq T_\ell^{(i)} \Delta_\ell \leq U_\ell$	(* Limits on template change *)

2.6 Evaluating Policies with Template Updates

We present a preliminary experimental evaluation of the ideas presented thus far using a prototype implementation.

Prototype Implementation: A prototype implementation was developed in Python, using the exact arithmetic LP solver QSOptEx. The QSOptEx solver provides a fast and convenient interface to an optimized Simplex implementation in exact arithmetic. Our implementation allows the specification of a transition system and supports a few additional features on top of those presented in the chapter including location invariants. We also support the option to specify different templates at various program locations. During the template update, our approach considers independent updates to the template at each location.

Specifying Template Changes: We consider a simple approach to specifying the limits L_ℓ, U_ℓ to the change in template at each location ℓ. First, the option for $\Delta_\ell = 0$ must be allowed, secondly, $\Delta_\ell = -T$ must be disallowed. For each $T_\ell(i,j) = 0$, we specify corresponding limits $L_\ell(i,j) = -z$ and $U_\ell(i,j) = z$ for a fixed constant $z > 0$ (taken as 1000 in our experiments). For $T_\ell(i,j) \neq 0$, we allow Δ to range between $\frac{1}{2}T_\ell(i,j)$ and $2T_\ell(i,j)$ in our experiments.

Benchmark Examples: We consider a set of 9 benchmark examples that are illustrative of applications that we encounter in the verification of discrete-time affine hybrid systems. Table 2.1 briefly describes each benchmark example.

Experimental Comparison: Table 2.2 shows the comparison between abstract interpretation using Kleene iteration and policy iteration with template update. Likewise, the performance for policy iteration without template update is shown in Table 2.3. Finally, Table 2.4 shows a comparison with the polyhedral abstract domain using the Parma Polyhedron Library (PPL) [BRZH02, BHZ06].

The table reports the objective value of the initial solution obtained after the Kleene iteration (using widening/narrowing) terminates. A non-negative value of the objective function indicates the failure to prove the property. Overall, we see that policy iteration with template update is

Table 2.1: Description of the benchmarks used and the sizes in terms of (# variables, # locations, # transitions)

ID	Size	Remark
1	(4,2,2)	Switched linear system with 4 state variables.
2	(2,2,4)	Example in Fig. 2.2.
3	(2,1,1)	Linear System with 1 location and transition.
4	(2,1,1)	Motivating example from Section 2.2.
5	(3,1,4)	Adjé et al. [AG15b].
6	(2,35,169)	Grid-based piecewise linearization of Van Der Pol oscillator.
7	(4,5,54)	A piecewise linear dynamical system.
8	(5,1,12)	Piecewise linear dynamical system.
9	(8,1,7)	A platoon of two cars with controller maintaining distance.

effective in these benchmarks in proving properties in 5 out of the 9 cases, whereas without template update we prove the property in just 2 out of 9. The polyhedral domain proves 3 out of the 9.

Figure 2.5 shows the sequence of iterates at the two locations for the transition system shown in Fig. 2.2 corresponding to benchmark number 2. The goal is to establish the unreachability of $x_2 \geq 0.8$ at location ℓ_2. The final invariant for ℓ_2 is shown in green, proving the specification.

Thus, we provide preliminary evidence that the bilinear approach is effective in cases where Kleene or policy iteration fail. At the same time, we notice that the size of the bilinear problem, though polynomial in the original transition system and template size, is often large with thousands of variables. However, the problems are sparse with each constraint involving just a tiny fraction of these variables. This points out the need for simplification techniques and approaches to solving bilinear problems that exploit this sparsity to make the approach more efficient.

Figure 2.5: Sequence of iterates for benchmark id 2 culminating in the final invariants shown shaded in blue and green. The property $x_2 \geq 0.8$ is shown unreachable at the green location by the final iterate.

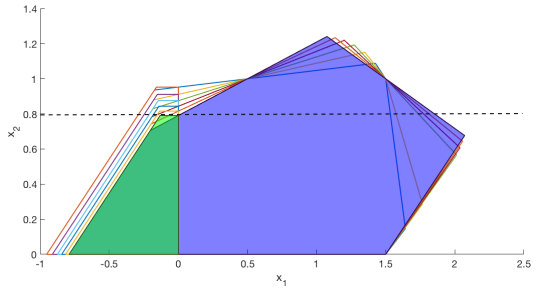


Table 2.2: Experimental results for policy iteration **with** template update. All experiments were run on a Macbook Air laptop with 1.8 GHz Intel processor, 8GB RAM running OSX10.12. All timings are in seconds. **Legend: Succ.**: whether the property was successfully proved, if not, the objective value is reported, |BOP|: size of the bilinear problem (# bilinear template variables, # bilinear mult. variables, # linear mult. variables), # I: # policy iterations - A (*) next to this number indicates that the iteration was stopped due to 5 consecutive steps with same objective value. **TO** indicates time out of 1 hour.

ID	Init. Templ.	Kleene		Policy Iteration			
	Type, $\|T\|$	Time	Succ.	|BOP|	Time	# I	Succ.
1	Interval, 8	0.12	N(0.2)	(240, 1176, 1249)	0.55	5(*)	N (0.2)
3	Octagon, 8	0.04	N(0.5)	(24, 72, 161)	0.05	2	Y
4	Interval, 4	0.02	N(15.5)	(12, 20, 33)	0.02	2	Y
5	Pentagon, 10	1.5	N(2.83)	(40, 410, 681)	0.3	2	Y
6	Interval, 4	2.5	N(0.75)	(168, 836, 2033)	2.9	5(*)	N(0.75)
7	Pentagon, 22	3.1	N(0.2)	(500, 10020, 8332)	2.4	3	Y
8	Pentagon, 22	2.6	N(43)	(126, 5313, 6311)	**TO**	-	N
9	Interval, 16	2.2	N(9500)	(286,4758,9501)	2.4	2	Y

Table 2.3: Experimental results for policy iteration **without** template update. See Table 2.2 for the legend.

ID	Init. Templ.	Kleene		Policy Iteration			
	Type, $\|T\|$	Time	Succ.	|BOP|	Time	# I	Succ.
1	Interval, 8	0.12	N (0.2)	(52,1176, 1249)	0.19	2	N(0.2)
2	Octagon, 8	0.15	N (0.2)	(16, 264, 353)	0.1	2	N(0.2)
3	Octagon, 8	0.04	N(0.5)	(8, 72, 161)	0.02	1	N(0.5)
4	Interval, 4	0.02	N(15.5)	(4,20,33)	0.01	1	N(15.5)
5	Pentagon, 10	1.5	N(2.83)	(10, 410, 681)	0.3	2	Y
6	Interval, 4	2.5	N(0.75)	(140, 836, 2033)	1.5	5(*)	N(0.75)
7	Pentagon, 22	3.1	N(0.2)	(110, 10020, 8322)	15.3	5(*)	N(0.2)
8	Pentagon, 22	2.6	N(43)	(22, 5313, 6311)	16.9	5(*)	N(38.8)
9	Interval, 16	2.2	N(9500)	(16, 4758, 9501)	2.3	2	Y

Table 2.4: Experimental results using the polyhedral abstract domain and comparison of outcome against policy iter with template change in column TC (recalled from Tab. 2.2).

ID	Time(s)	Succ.	TC
1	0.6	N	N
2	0.03	Y	Y
3	0.02	Y	Y
4	0.02	Y	Y
5	0.3	N	**Y**
6	1.7	N	N
7	2.7	N	**Y**
8	TO $> 1h$	N	N
9	TO $> 1h$	N	**Y**

Chapter 3

Bilinear Programming Problems

Thus far, we have studied the problem of abstract interpretation of programs using the template domain. As noted earlier, the data flow constraints characterizing the fixed points of the problem correspond to a class of non-convex optimization problems known as **bilinear programs** (BLPs). In this chapter, we study the basic structure of the bilinear programs, abstracting from the structure of the optimization problem presented in Figure 2.4, and propose two novel approaches to solving bilinear programs that naturally handle some of their inherent difficulties. We discuss the connection between bilinear and quadratic programs (QPs) to motivate the use of QP solvers on BLPs. We conclude by comparing our novel approaches against a suite of varying methods on both real-world and synthetic data. We discuss the applicability of each method and compare their ability to accurately solve bilinear programs.

Our problem of interest is the following generic Bilinear Program (BLP):

$$\begin{array}{lll} \min\limits_{\vec{x},\vec{y},\vec{z}} & \vec{x}^T A_0 \vec{y} + \vec{b_0}^T \vec{x} + \vec{c_0}^T \vec{y} + \vec{d_0}^T \vec{z} & \\ \text{subject to} & \vec{x}^T A_i \vec{y} + \vec{b_i}^T x + \vec{c_i}^T y + \vec{d_i}^T \vec{z} \leq \vec{r}_i \quad i \in \mathcal{I} & \\ & \vec{x}^T A_j \vec{y} + \vec{b_j}^T \vec{x} + \vec{c_j}^T \vec{y} + \vec{d_j}^T \vec{z} = \vec{r}_j \quad j \in \mathcal{E} & \\ & \vec{l}_x \leq \vec{x} \leq \vec{u}_x,\ \vec{l}_y \leq \vec{y} \leq \vec{u}_y,\ \vec{l}_z \leq \vec{z} \leq \vec{u}_y & \end{array} \tag{3.1}$$

where $\vec{x} \in \mathbb{R}^n$, $\vec{y} \in \mathbb{R}^m$, $\vec{z} \in \mathbb{R}^k$ and $\vec{l}_x, \vec{u}_x, \vec{l}_y, \vec{u}_y, \vec{l}_z, \vec{u}_z$ represent lower and upper bounds on $\vec{x}, \vec{y}$ and $\vec{z}$, respectively. $\mathcal{I}$ and $\mathcal{E}$ are the inequality and equality index sets for the constraints, respectively.

The bilinear program in (3.1) is **separable**. By fixing the variables $\vec{x}$ to definite values, we

obtain a linear program over the remaining variables $(\vec{y}, \vec{z})$, and likewise, fixing the variables $\vec{y}$, we obtain a LP over $(\vec{x}, \vec{z})$.

Lemma 3.0.1. *The dataflow constraints obtained in Figure 2.4 form a bilinear program (BLP) with the variables $\vec{x}$ denoting the template variables Δ_ℓ and $\vec{c}(\ell)$ for $\ell \in \mathcal{L}$ and the $\vec{y}$ denoting the "bilinear multiplier variables" Λ_τ and $\vec{z}$ representing the "linear multiplier variables" $\vec{\lambda}_s, \Lambda_0, \Gamma_\tau$.*

It is easy to see that the feasible region for a BLP is nonconvex in general. Also, finding if a BLP is feasible is NP-hard.

Theorem 3.0.2. *Checking if there exists $(\vec{x}, \vec{y}, \vec{z}) \in \mathbb{R}^{n+m+k}$ that satisfy the constraints of a BLP* (3.1) *is NP-hard.*

Proof. We reduce from 3-CNF SAT problem that consists of n Boolean variables $p_1, \ldots, p_n$ and m clauses, each clause A_i consisting of a subset of literals $A_i \subseteq \{p_1, \ldots, p_n, \overline{p_1}, \ldots, \overline{p_n}\}$.

We introduce variables x_i, y_i corresponding to each proposition p_i, with the goal that $x_i = 1, y_i = 0$ will denote assigning p_i to *true* and $x_i = 0, y_i = 1$ will denote assigning p_i to *false*. To ensure that these are the only possible values, we write the constraints

$$\begin{aligned} x_i y_i &\leq 0, \\ x_i + y_i &= 1, \\ x_i &\in [0,1] \\ y_i &\in [0,1] \end{aligned} \tag{3.2}$$

Note that the constraint $x_i y_i \leq 0$ is a bilinear inequality. The remaining are linear inequalities. Also, note that the only feasible solutions are $(x_i, y_i) \in \{(0,1), (1,0)\}$.

Next, for each clause A_j involving some literals, we add the constraint

$$\sum_{p_i \in A_j} x_i + \sum_{\overline{p_i} \in A_j} y_i \geq 1\,. \tag{3.3}$$

Now consider the system of constraints obtained by collecting (3.2) for $i \in \{1, \ldots, n\}$ and (3.3) for $j \in \{1, \ldots, m\}$. We can verify that the system forms the constraints for a BLP with $\vec{x} : (x_1, \ldots, x_n)^T$, $\vec{y} : (y_1, \ldots, y_n)^T$ and $\vec{z}$ as the empty vector.

Next, we note that for any satisfying assignment to the original problem, the corresponding assignment to (x_i, y_i) as described above will satisfy the constraints for the BLP. Similarly, we have proven that if the BLP is satisfiable, then for each (x_i, y_i), we can set the corresponding proposition $p_i = \mathit{true}$ if $x_i = 1$ and *false* otherwise. Due to constraint (3.3), each clause has at least one literal that will be true.

This completes the reduction and the proof of NP-hardness. □

3.1 Policy Iteration: Alternating Minimization

We have already observed that if we could fix $\vec{x}$ to concrete values in the BLP (3.1), we obtain an LP in terms of the remaining variables $(\vec{y}, \vec{z})$ and likewise for $\vec{y}$. A simple strategy is to perform alternating minimization wherein, we find an initial feasible solution $(\vec{x}_0, \vec{y}_0, \vec{z}_0)$ and improve the current solution $(\vec{x}_i, \vec{y}_i, \vec{z}_i)$ at each iteration by carrying out two steps: (a) Fixing $\vec{x} : \vec{x}_i$, solve the resulting LP to obtain $(\vec{y}_{i+1}, \vec{z})$; and (b) fixing $\vec{y} : \vec{y}_{i+1}$, solve the resulting LP to obtain $(\vec{x}_{i+1}, \vec{z}_{i+1})$. Combining the two steps, we obtain the next solution $(\vec{x}_{i+1}, \vec{y}_{i+1})$. We continue this process until no more improvements are possible. There are two basic questions posed by this algorithm: (a) does it terminate? and (b) if it terminates, then how does the result compare with the optimal solution to the problem?

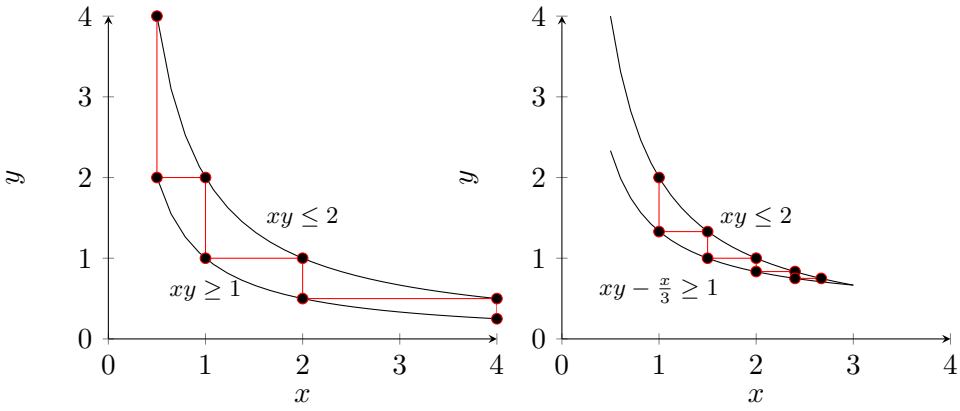


Figure 3.1: Examples showing nontermination of alternating minimization.

First, it is easy to formulate an example that illustrates non-termination.

Nontermination of Alternating Minimization Consider the problem below illustrated in Figure 3.1 (left).

$$\begin{array}{ll} \max & x - y \\ & xy \quad \geq 1 \\ & xy \quad \leq 2 \\ & x, y \quad \geq 0 \end{array} \tag{3.4}$$

Starting with the initial solution $(x_0 = 1, y_0 = 1)$, we note that the sequence of iterates obtained using alternating minimization are

$$(x_1 = 2, y_1 = \frac{1}{2}), (x_2 = 4, y_2 = \frac{1}{4}), \cdots, (x_j = 2^j, y_j = 2^{-j}) \cdots$$

The process continues forever without termination with each LP producing a bounded result although the original BLP is unbounded.

In fact, alternating minimization may fail to terminate even if the feasible region of the BLP is bounded.

Nontermination of Alternating Minimization: Bounded Case Consider the problem below illustrated in Figure 3.1 (right).

$$\begin{array}{ll} \max & x - y \\ & xy \quad \leq 2 \\ & x(y - \frac{1}{3}) \quad \geq 1 \\ & x, y \quad \geq 0 \end{array} \tag{3.5}$$

The reader may verify that starting with $(x_0 = 1, y_0 = 1)$, the approach iterates to obtain $(x_{i+1} = \frac{6x_i}{3+x_i},\ y_{i+1} = \frac{3+x_i}{3x_i})$. As $i \to \infty$, the process converges to $x^* = 3, y^* = \frac{2}{3}$, which can be verified as the optimal solution to the problem by graphing the constraints.

Finally, we illustrate so-called saddle points, wherein the process converges $(x_{i+1}, y_{i+1}) = (x_i, y_i)$, but the solution is not a KKT point (a local optimum) of the original BLP.

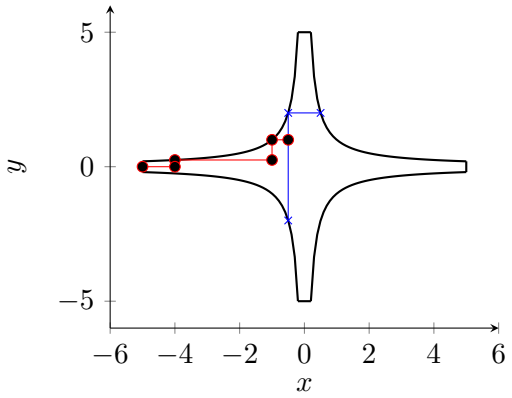


Figure 3.2: BLP with saddle point at $(-1, 1)$. The red path shows how an interior point solver can solve the problem whereas the simplex approach shown in blue can be stuck in a saddle point.

Saddle Points Consider the problem instance shown below illustrated in Figure 3.2.

$$\begin{aligned} \min_{x,y} \quad & x \\ \text{subject to} \quad & xy \leq 1 \\ & xy \geq -1 \\ & -5 \leq x \leq 5 \\ & -5 \leq y \leq 5 \end{aligned} \tag{3.6}$$

The point $(-1, 1)$ is a saddle point of the problem. Fixing $x = -1$ yields the optimization problem

$$\min\ 0 \text{ s.t. } -1 \leq y \leq 1\,, \tag{3.7}$$

which can yield either $y = -1$ or $y = 1$ if a Simplex-based LP solver is used. Assuming that the solver results in $y = 1$, we obtain no further change in the values (x, y). If $y = -1$ were chosen by the solver, instead, the very next step yields the desired saddle point. The point $(-1, 1)$ is not a local optimum for the problem. For instance the descent direction $(\Delta x, \Delta y) : (-1, 1)$ can maintain feasibility while providing a step that decreases the objective value. Unfortunately, alternating minimization is unable to find the descent direction since it is restricted to finding directions with $\Delta x = 0$ or $\Delta y = 0$. Unfortunately, such directions do not exist at the saddle point.

We also note that if a simplex solver were not used, it is theoretically possible that an "interior point" solver can choose $y = 0$ as the solution to the LP (3.7). Subsequently, we can solve for x and obtain a globally optimal solution.

In the context of saddle point, the key problem is that of moving the iteration past the saddle point to continue improving the solution? This problem has remained mostly open thus far. We summarize some of the difficulties encountered. First, we note that alternating minimization is a form of **coordinate descent** wherein the solution improvement direction for the problem in (3.1), wherein starting from a current solution $(\vec{x}, \vec{y}, \vec{z})$ that is feasible, we seek a new solution $(\vec{x} + \Delta_x, \vec{y} + \Delta_y, \vec{z} + \Delta_z)$, such that (a) $(\vec{x} + \Delta_x, \vec{y} + \Delta_y, \vec{z} + \Delta_z)$ is feasible and

$$f(\vec{x} + \Delta_x, \vec{y} + \Delta_y, \vec{z} + \Delta_z) < f(\vec{x}, \vec{y}, \vec{z}) \,,$$

wherein $f(\vec{x}, \vec{y}, \vec{z}) : \ \vec{x}^T C_0 \vec{y} + \vec{a}_0^T \vec{x} + \vec{b}_0^T \vec{y} + \vec{c}_0^T \vec{z}$ is the objective function shown for the problem (3.1). However, it is immediate that the search for such feasible directions $(\Delta_x, \Delta_y, \Delta_z)$ requires solving a bilinear problem with a similar structure as the original problem (3.1) in the first place. Thus, linearity is forced through coordinate descent wherein either $\Delta_x = 0$ or $\Delta_y = 0$. Thus, the problem of "escaping" from local saddle points is an open problem that limits alternating minimization approaches.

3.2 Bilinear Problems and QCQPs

There exists a natural encoding of BLPs to Quadratically-Constrained Quadratic Programs (QCQPs) of a higher dimensional space. That is, any BLP can be reformulated as the QCQP and numerous approaches to solving QCQPs can be applied to BLPs, as well. Specifically, we can

encode (3.1) as the QCQP

$$\begin{aligned} \min_{\tilde{\vec{x}}} \quad & \tilde{\vec{x}}^T \widetilde{A}_0 \tilde{\vec{x}} + \tilde{\vec{b}}_0^T \tilde{\vec{x}} \\ \text{subject to} \quad & \tilde{\vec{x}}^T \widetilde{A}_i \tilde{\vec{x}} + \tilde{\vec{b}}_i^T \tilde{\vec{x}} \leq \vec{r}_i \quad i \in \mathcal{I} \\ & \tilde{\vec{x}}^T \widetilde{A}_i \tilde{\vec{x}} + \tilde{\vec{b}}_i^T \tilde{\vec{x}} = \vec{r}_i \quad i \in \mathcal{E} \\ & \tilde{\vec{l}} \leq \tilde{\vec{x}} \leq \tilde{\vec{u}} \end{aligned} \tag{3.8}$$

where $\tilde{\vec{x}} = \begin{pmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{pmatrix} \in \mathbb{R}^{m+n}$, $\widetilde{A}_0 = \frac{1}{2} \begin{pmatrix} 0 & A_0 & 0 \\ A_0^T & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, $\tilde{\vec{b}}_0 = \begin{pmatrix} \vec{b}_0 \\ \vec{c}_0 \\ \vec{d}_0 \end{pmatrix}$, $\tilde{\vec{l}} = \begin{pmatrix} \vec{l}_x \\ \vec{l}_y \\ \vec{l}_z \end{pmatrix}$, $\tilde{\vec{u}} = \begin{pmatrix} \vec{u}_x \\ \vec{u}_y \\ \vec{u}_z \end{pmatrix}$, and $\forall i \in \mathcal{I} \cup \mathcal{E}$, $\widetilde{A}_i = \frac{1}{2} \begin{pmatrix} 0 & A_i & 0 \\ A_i^T & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, $\tilde{\vec{b}}_i = \begin{pmatrix} \vec{b}_i \\ \vec{c}_i \\ \vec{d}_i \end{pmatrix}$. Note that in the QCQP, the matrices $\widetilde{A}_0$ and $\widetilde{A}_i$ are symmetric, whereas the matrices A_0 and A_i in the BLP may not even be square.

Conversely, for any QCQP there exists an encoding into a BLP. We can see this by replacing all terms of the form $\tilde{\vec{x}}^T \widetilde{A}_i \tilde{\vec{x}}$ with $\tilde{\vec{x}}^T \widetilde{A}_i \tilde{\vec{y}}$ and embedding the constraint $\tilde{\vec{x}} = \tilde{\vec{y}}$ into the problem. However, the resulting BLP has equality constraints $\tilde{\vec{x}} = \tilde{\vec{y}}$. Although it seemingly maintains the "separability" property, it is immediately clear that solving it via policy iteration (also referred to as "alternating minimization" in this section) is useless; that is, if we fix $\tilde{\vec{x}}$ and solve for $\tilde{\vec{y}}$, the constraint $\tilde{\vec{x}} = \tilde{\vec{y}}$ means the method stagnates.

Rewriting the BLP as a QCQP in order to solve it numerically is not always a good idea, since the new QCQP is much larger, but it motivates us to consider some particular QCQP methods and their application to BLPs. Notice that there is no other requirement on $\widetilde{A}_i$ other than symmetry.

If we further assume that each $\widetilde{A}_i$ is positive semidefinite for $i \in \{0\} \cup \mathcal{I}$, and $\widetilde{A}_i = 0$ for $i \in \mathcal{E}$, then the problem is convex. This is generally not the case for problems arising from program analysis, and therefore, we do not assume our QCQPs are convex.

Because neither the general QCQPs nor BLPs are convex, most nonlinear solvers can only produce a stationary point, which may be either a local minimizer (and possibly a global minimizer)

or a saddle point. While ideally one wants a global rather than local minimizer, a secondary goal is to find a local minimizer rather than a saddle point. We focus on this second problem, and note that it has been the subject of much recent attention in machine learning, where it is sometimes argued that all local minimizers are "good enough" and thus one only needs to avoid saddle points [DPG$^+$14].

Various relaxations for solving nonconvex QPs have been proposed including lift-and-project cutting plane algorithms, reformulation-linearization techniques (RLT), semi-definite programming (SDP) relaxations, matrix-cut algorithms, and semi-infinite linear programs [AHJS00, Lin05, QBM12, ZSL11, SF02, PB17].

Convex Relaxation Techniques: Relaxations for QCQPs based on SDPs or RLTs lift the original problem into a higher dimension by implementing a nonlinear change of variables that puts all the nonconvexity into a single constraint, and then drops the nonconvex constraint in order to relax the original problem to a convex program. Both the SDP and RLT relaxations are invariant to an invertible affine transformation of the original variables. One of the commonly used change of variables is $\vec{x} \to \vec{x}\vec{x}^T = X$, lifting the problem from $\vec{x} \in \mathbb{R}^n$ to $X \in \mathbb{S}^n_+$, and replacing quadratic terms with matrix inner products using the trace operator: $\vec{x}^T A \vec{x} = \text{Tr}(\vec{x}^T A \vec{x}) = \text{Tr}(A\vec{x}\vec{x}^T) = A \cdot X$.

$$\begin{aligned} \min_{\tilde{\vec{x}}, X} \quad & \widetilde{A}_0 \cdot X + \tilde{\vec{b}}_0^T \tilde{\vec{x}} \\ \text{subject to} \quad & \widetilde{A}_i \cdot X + \tilde{\vec{b}}_i^T \tilde{\vec{x}} \le \vec{r}_i \quad i \in \mathcal{I} \\ & C_i \cdot X + \tilde{\vec{b}}_i^T \tilde{\vec{x}} = \vec{r}_i \quad i \in \mathcal{E} \\ & X = \tilde{\vec{x}}\tilde{\vec{x}}^T \\ & \tilde{\vec{l}} \le \tilde{\vec{x}} \le \tilde{\vec{u}} \end{aligned} \tag{3.9}$$

Along with the constraints $X = \tilde{\vec{x}}\tilde{\vec{x}}^T$, this reformulation does not change the problem, and is still nonconvex, though all terms except the $X = \tilde{\vec{x}}\tilde{\vec{x}}^T$ constraint are linear. To get a relaxation, the equality constraint $X = \tilde{\vec{x}}\tilde{\vec{x}}^T$ is relaxed to $X \succeq \tilde{\vec{x}}\tilde{\vec{x}}^T$ where $\succeq$ is the Loewner ordering where $Y \succeq 0$ means Y is a symmetric positive semi-definite matrix. This new inequality defines a convex set, and the relaxation is a convex program and in particular a SDP.

The reformulation-linearization technique follows the same procedure of employing auxiliary variables in place of the product terms, $\tilde{\vec{x}}_i\tilde{\vec{x}}_j = X_{ij}$, but rather than imposing a positive semidefinite constraint, RLTs add the product of bound-factors $(\tilde{\vec{u}}_i - \tilde{\vec{x}}_i)(\tilde{\vec{x}}_j - \tilde{\vec{l}}_j) \geq 0$ for any or all pairs of variables $\tilde{\vec{x}}_i,\ i \in \{1, \ldots, n\}$, where $\tilde{\vec{u}}_i$ and $\tilde{\vec{l}}_j$ denote the *finite* upper and lower bounds on variables $\tilde{\vec{x}}_i$ and $\tilde{\vec{x}}_j$, respectively. Furthermore, RLT constraints can include the products of equality constraints with monomials $\Pi_j\tilde{\vec{x}}_j$ and inequality constraints with themselves as well as the bound-factors mentioned above. While these extra constraints may be redundant, after a round of reformulation and linearization they lead to a tighter convex relaxation.

Previous work on combining RLTs and SDP relaxations to remove a substantial portion of the feasible region was explored in [Ans09] with provable results. Sherali et al. [SF02] investigated the effects of enhancing the RLT formulations with semi-definite cuts on QPs without quadratic constraints. They verify empirically that combining certain aspects from each formulation leads to better results. Other convex relaxations for specific problem structures, including Second-Order Conic Programs, have been proposed to speed up computation while preserving accuracy [KK01, SDL12].

In addition to SDP and RLT relaxations, McCormick Envelopes provide a convex relaxation for bilinear programs commonly used to solve Mixed Integer NonLinear Programs (MINLPs). They are designed to guarantee convexity while keeping the bounds on the original problem sufficiently tight. The method behind McCormick Envelopes is to replace distinct products of variables, i.e. bilinear terms, with auxiliary variables and append bound constraints which form convex over- and under-estimators of the bilinear terms. The essence of the method is similar to that of the reformulation-linearization technique; however, the goal of McCormick Envelopes is to replace all bilinear terms with appropriately bounded auxiliary variables whereas the goal of RLTs is to append additional, potentially redundant, constraints in order to minimize the set of candidate solutions to the relaxed problem.

In the context of proving program properties, relaxation-based approaches are less useful since they focus on proving lower bounds for the optimal value of a minimization problem. Therefore,

in the setting of this chapter, it is possible to use relaxation methods to establish that no linear invariant involving a fixed number of conjunctions can prove a property. However, unless the relaxation is exact, these methods do not directly find invariants to establish a property since the solution obtained by the relaxation may not be feasible.

3.3 Augmented Lagrangian Approaches

Augmented Lagrangian methods are a promising approach to solving constrained optimization problems through a series of related unconstrained problems [NW06]. The constraints are incorporated into the objective function via a quadratic penalty plus a pseudo-Lagrangian term. By increasing the effect of the penalty parameter, the sequence of iterates tend to a local solution of the original constrained problem.

We now present our novel method to solve BLPs using the augmented Lagrangian framework. Augmented Lagrangian methods naturally handle equality constraints. Since one of the intrinsic difficulties in solving BLPs arises from the restrictive nature of their equality constraints, we take advantage of the method's ability to transform a constrained optimization problem into a sequence of unconstrained problems and specialize it to exploit the structure of BLPs. To handle inequality constraints, there are several variants, and we follow the variant that converts inequality constraints into equality constraints via a slack variable s, but keeps all box constraints (including those on the slack) explicit. We reformulate (3.1) as follows

$$\begin{aligned} \min_{\vec{x},\vec{y},\vec{s}} \quad & \underbrace{\vec{x}^T A_0 \vec{y} + \vec{b}_0^T \vec{x} + \vec{c}_0^T \vec{y}}_{f(\vec{x},\vec{y})} \\ \text{subject to} \quad & \underbrace{\vec{x}^T A_i \vec{y} + \vec{b}_i^T \vec{x} + \vec{c}_i^T \vec{y} - r_i}_{g_i(\vec{x},\vec{y})} + s_i = 0 \quad i \in \mathcal{I} \cup \mathcal{E} \\ & \vec{l}_x \le \vec{x} \le \vec{u}_x, \quad \vec{l}_y \le \vec{y} \le \vec{u}_y, \\ & s_i \ge 0 \ \forall i \in \mathcal{I}, \quad s_i = 0 \ \forall i \in \mathcal{E} \end{aligned} \tag{3.10}$$

For convenience, the $\vec{z}$ variables are assumed to be nonexistent in our exposition. However, the

approach described can be readily extended to include linear $\vec{z}$ variables, as well. We will denote the box constraints concisely as $(\vec{x}, \vec{y}, \vec{s}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{S}$. Since $g_i(\vec{x}, \vec{y}) + s_i = 0$ for any feasibly points, we can see that for any $\mu > 0$, (3.10) is equivalent to

$$\begin{aligned} \min_{(\vec{x},\vec{y},\vec{s}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{S}} \quad & f(\vec{x}, \vec{y}, \vec{z}) + \frac{\mu}{2} \sum_{i \in \mathcal{E} \cup \mathcal{I}} (g_i(\vec{x}, \vec{y}, \vec{z}) + s_i)^2 \\ \text{subject to} \quad & g_i(\vec{x}, \vec{y}, \vec{z}) + s_i = 0 \quad i \in \mathcal{I} \cup \mathcal{E} \end{aligned} \tag{3.11}$$

The augmented Lagrangian method then applies subgradient ascent with respect to the dual variable $\vec{\lambda}$ on the Lagrangian of (3.11). Starting at some $\vec{\lambda}^0 \in \mathbb{R}^{|\mathcal{I} \cup \mathcal{E}|}$, and defining the augmented Lagrangian $\mathcal{L}_A$ as

$$\mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}, \mu) \stackrel{\text{def}}{=} f(\vec{x}, \vec{y}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i \cdot (g_i(\vec{x}, \vec{y}) + s_i) + \frac{\mu}{2} \sum_{i \in \mathcal{E} \cup \mathcal{I}} (g_i(\vec{x}, \vec{y}) + s_i)^2$$

the method iterates for $k = 1, 2, \ldots,$

$$(\vec{x}^k, \vec{y}^k, \vec{s}^k) \leftarrow \operatorname*{argmin}_{(\vec{x},\vec{y},\vec{s}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{S}} \mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k) \tag{3.12}$$

$$\vec{\lambda}_i^{k+1} \leftarrow \vec{\lambda}_i^k - \mu^k \left(g_i(\vec{x}^k, \vec{y}^k) + s_i^k \right) \tag{3.13}$$

$$\mu^{k+1} \leftarrow \rho \mu^k \tag{3.14}$$

where $\rho > 1$.

While the augmented Lagrangian method has been applied generically to non-convex nonlinear programs, as in the `PENNON` package [KS03], here we can specialize it to take advantage of the structure of BLP problems. Note that (3.12) can be written as

$$\min_{\vec{x} \in \mathcal{X}} \underbrace{\left(\min_{(\vec{y},\vec{s}) \in \mathcal{Y} \times \mathcal{S}} \mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k) \right)}_{\varphi(\vec{x})} \tag{3.15}$$

and evaluating $\varphi(\vec{x})$—that is, finding the minimizing points (y_x, s_x)—requires solving a **convex** quadratic program, which can be done efficiently (our implementation uses the `MATLAB` software `CVX` and `Gurobi`). Furthermore, $\nabla\varphi$ can be calculated using the following lemma. This lemma is similar to Danskin's Theorem [BNO03] but the proof is distinct since we minimize rather than maximize, and do not assume compact constraints.

Lemma 3.3.1. *Assume* $(\vec{y}_x, \vec{s}_x)$ *are the unique minimizers of* $\mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k)$ *over* $(\vec{y}, \vec{s}) \in \mathcal{Y} \times \mathcal{S}$ *and* $\vec{l}_x, \vec{u}_x, \vec{l}_y, \vec{u}_y$ *are real-valued (i.e, they do not have* $\pm\infty$ *entries) . Then* $\nabla\varphi(\vec{x}) = \nabla_x \mathcal{L}_A(\vec{x}, \vec{y}_x, \vec{s}_x; \vec{\lambda}^k, \mu^k)$.

Proof. The main tool is Thm. 10.58 in [RW09]. Let $\varphi(\vec{x}) = \min_{(\vec{y},\vec{s})} \mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k)$ and $\Phi(\vec{x}) = \operatorname{argmin}_{(\vec{y},\vec{s})} \mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k)$ denote the optimal value and the optimal solution set, respectively. Since $\mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k)$ is continuous, proper, and level-set bounded in $(\vec{y}, \vec{s})$ uniformly locally in $\vec{x}$, $\varphi(\vec{x})$ is proper, and continuous and for each $\vec{x} \in \operatorname{dom}(\varphi)$ the set of minimizers $\Phi(\vec{x}) = \{(\vec{y}_x, \vec{s}_x)\}$ is nonempty and compact. Moreover, there exists a compact neighborhood B of $\vec{x}$ such that the image $\vec{x}$ under Φ is a compact set for all $\vec{x} \in B$. For each $(\vec{y}, \vec{s}) \in \Phi(B)$ the function $\mathcal{L}_{A(\vec{y},\vec{s})} := \mathcal{L}_A(\cdot, \vec{y}, \vec{s})$ is continuously differentiable on the interior of B. Let $\mathcal{O} := \operatorname{int}(B)$. Then $\varphi(\vec{x}) = \min_{(\vec{y},\vec{s})\in\mathcal{Y}\times\mathcal{S}} \mathcal{L}_{A(\vec{y},\vec{s})}(\vec{x})$ is continuously differentiable on $\mathcal{O}$.

□

Level-set bounded in $(\vec{y}, \vec{s})$ uniformly locally in $\vec{x}$ means that for each $\vec{x} \in \mathcal{X}$ and $\alpha \in \mathbb{R}$ there is a neighborhood V of $\vec{x}$ and bounded set $B \subset \mathcal{Y}\times\mathcal{S}$ such that the level-sets $\{(\vec{y}, \vec{s}) | \mathcal{L}_A(\vec{x}, \vec{y}, \vec{s}; \vec{\lambda}^k, \mu^k) \leq \alpha\} \subset B$ for all $\vec{x} \in V$, that is, the level-sets are bounded. The uniform level boundedness assumption allows us to find a neighborhood V of $\vec{x}$ satisfying the criteria for $\varphi(\vec{x})$ to be lower semi-continuous which is a weaker statement than continuity, but sufficient for the original proof of Thm. 10.58 in [RW09].

Now that we have a means to calculate $\nabla\varphi(\vec{x})$, we can minimize φ over $\vec{x} \in \mathcal{X}$ using gradient-based method such as L-BFGS-B [BLN95]. Our motivation for pursuing gradient-based approaches, rather than alternating minimization approaches, is due to a very recent body of literature showing that gradient descent is (under certain conditions) "unlikely" to be trapped at saddle points. In particular, Jin et al [JGN$^+$17] show that small infrequent random perturbations are sufficient to prevent gradient descent from converging to a saddle point, while [LSJR16] shows that random initializations prevent convergence to a saddle point. This formulation allows for bilinear optimization where $\vec{x}$ and $\vec{y}$ can be treated separately without the disjoint treatment of alternating minimization.

3.4 Quadratic Optimization and the S-Lemma

Although we potentially risk encountering many local optima when solving non-convex programs, there exist special cases of these programs for which exact relaxations exist. Quadratic programs involving a single constraint are one such example. Consider the problem

$$\begin{aligned} \min_{\vec{x}} \quad & \vec{x}^T A_0 \vec{x} + 2\vec{b_0}^T \vec{x} + \vec{c}_0 \\ \text{subject to} \quad & \vec{x}^T A_1 \vec{x} + 2\vec{b_1}^T \vec{x} + \vec{c}_1 \leq 0 \end{aligned} \tag{3.16}$$

where $\vec{x} \in \mathbb{R}^n$, $A_i \in \mathbb{S}^n$, $\vec{b}_i \in \mathbb{R}^n$, and $\vec{c}_i \in \mathbb{R}$. We do not require either quadratic to be convex, that is, $A_0, A_1 \not\succeq 0$ necessarily. This potentially non-convex program can still be solved efficiently using a semidefinite programming relaxation. The following celebrated result, known as the S-lemma [PT07], is crucial to solving the single constraint QCQP and allows us to certify the existence of a solution.

Theorem 3.4.1. *(S-Lemma, [Yak71]) Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be quadratic functions such that $g(\overline{\vec{x}}) > 0$ for some $\overline{\vec{x}} \in \mathbb{R}^n$. If for every $\vec{x}$, $(g(\vec{x}) \geq 0) \implies (f(\vec{x}) \geq 0)$, i.e., f and g are copositive, then there exists a λ such that $f(\vec{x}) \geq \lambda g(\vec{x})$ for all $\vec{x}$.*

Here, λ can be thought of as a certificate for the implication. Noting that

$$\begin{aligned} \min_{\vec{x}} \quad & \vec{x}^T A_0 \vec{x} + 2\vec{b_0}^T \vec{x} + \vec{c}_0 \\ \text{subject to} \quad & -(\vec{x}^T A_1 \vec{x} + 2\vec{b_1}^T \vec{x} + \vec{c}_1) \geq 0 \end{aligned} \tag{3.17}$$

is equivalent to

$$\begin{aligned} \max_{\gamma} \quad & \gamma \\ \text{subject to} \quad & (-(\vec{x}^T A_1 \vec{x} + 2\vec{b_1}^T \vec{x} + \vec{c}_1) \geq 0) \implies (\vec{x}^T A_0 \vec{x} + 2\vec{b_0}^T \vec{x} + \vec{c}_0 - \gamma \geq 0), \end{aligned} \tag{3.18}$$

we can use the S-lemma to rewrite (3.16) as an SDP. That is, (3.18) can be rewritten as

$$\begin{aligned} \max_{\gamma,\lambda} \quad & \gamma \\ \text{subject to} \quad & \vec{x}^T A_0 \vec{x} + 2\vec{b_0}^T \vec{x} + \vec{c}_0 - \gamma \geq -\lambda(\vec{x}^T A_1 \vec{x} + 2\vec{b_1}^T \vec{x} + \vec{c}_1), \ \forall \vec{x} \\ & \lambda \geq 0, \end{aligned} \tag{3.19}$$

which is equivalent to

$$\begin{aligned} \max_{\gamma,\lambda} \quad & \gamma \\ \text{subject to} \quad & \lambda \geq 0 \\ & \begin{pmatrix} A_0 + \lambda A_1 & \vec{b}_0 + \lambda \vec{b}_1 \\ (\vec{b}_0 + \lambda \vec{b}_1)^T & \vec{c}_0 + \lambda \vec{c}_1 - \gamma \end{pmatrix} \succeq 0, \end{aligned} \tag{3.20}$$

an SDP in variables γ and λ.

We could have arrived at the same result using the Lagrangian dual function:

$$\begin{aligned} g(\lambda) &= \inf_{\vec{x}} \mathcal{L}(\vec{x}, \lambda) \\ &= \inf_{\vec{x}} \vec{x}^T (A_0 + \lambda A_1)\vec{x} + 2(\vec{b}_0 + \lambda \vec{b}_1)^T \vec{x} + \vec{c}_0 + \lambda \vec{c}_1 \\ &= \begin{cases} \vec{c}_0 + \lambda \vec{c}_1 - (\vec{b}_0 + \lambda \vec{b}_1)^T (A_0 + \lambda A_1)^\dagger (\vec{b}_0 + \lambda \vec{b}_1), & A_0 + \lambda A_1 \succeq 0, \ \vec{b}_0 + \lambda \vec{b}_1 \in \mathcal{R}(A_0 + \lambda A_1) \\ -\infty & \text{otherwise} \end{cases} \end{aligned} \tag{3.21}$$

Using the Schur complement we derive the same SDP program as above

$$\begin{aligned} \max_{\gamma,\lambda} \quad & \gamma \\ \text{subject to} \quad & \lambda \geq 0 \\ & \begin{pmatrix} A_0 + \lambda A_1 & \vec{b}_0 + \lambda \vec{b}_1 \\ (\vec{b}_0 + \lambda \vec{b}_1)^T & \vec{c}_0 + \lambda \vec{c}_1 - \gamma \end{pmatrix} \succeq 0. \end{aligned} \tag{3.22}$$

If Slater's condition is satisfied, meaning the interior of the feasible region is non-empty, we have strong duality and the optimal values of the primal and dual problems are equivalent.

Even when the primal problem is not convex, the dual function g is always a concave function of λ. Furthermore, the dual of the dual function is always convex, and under certain conditions, including convexity, is equal to the primal problem. When the primal problem is not convex, then the dual of the dual program may serve as a convex relaxation to the original problem.

The dual of the SDP (3.22) is

$$\begin{aligned} \min_{\vec{x}} \quad & \mathbf{Tr}(A_0 X) + 2\vec{b_0}^T\vec{x} + \vec{c}_0 \\ \text{subject to} \quad & \mathbf{Tr}(A_1 X) + 2\vec{b_1}^T\vec{x} + \vec{c}_1 \leq 0 \\ & \begin{pmatrix} X & \vec{x} \\ \vec{x}^T & 1 \end{pmatrix} \succeq 0, \end{aligned} \tag{3.23}$$

or

$$\begin{aligned} \min_{\vec{x}} \quad & \mathbf{Tr}(A_0 X) + 2\vec{b_0}^T\vec{x} + \vec{c}_0 \\ \text{subject to} \quad & \mathbf{Tr}(A_1 X) + 2\vec{b_1}^T\vec{x} + \vec{c}_1 \leq 0 \\ & X - \vec{x}\vec{x}^T \succeq 0, \end{aligned} \tag{3.24}$$

an SDP with $X \in \mathbb{S}^n$ and $\vec{x} \in \mathbb{R}^n$, where we again use the Schur complement. Moreover, by replacing the convex term $X - \vec{x}\vec{x}^T \succeq 0$ with $X = \vec{x}\vec{x}^T$, we arrive at the original program (3.16). However, the objective is now a linear function in variables $\vec{x}$ and X and the non-convexity of the original problem is captured in the constraint $X = \vec{x}\vec{x}^T$. Clearly, (3.24) is a relaxation of the original problem since it allows for a larger feasible set of solutions by relaxing the rank-1 constraint to a positive semidefinite one.

Since the constraint relaxation means we minimize the same objective over a larger set of feasible solutions, the relaxation provides a lower bound for the original problem. By duality, if Slater's condition holds for (3.16) then strong duality holds between the original problem and its convex relaxation. Thus, the relaxation (3.24) is exact provided the interior of the feasible region is non-empty.

While this method is exact for problems of the form (3.8) with $|\mathcal{I}| = 1$, $\mathcal{E} = \{\}$, and unbounded variable $\vec{x}$, our space of feasible solutions may be unbounded; thus, we are not guaranteed a bounded

objective value. However, its implementation still allows us to deduce a lower bound for the original problem when the global optimal value is unknown. For this reason, we include it in our list of solvers.

3.5 Projected Gradient Descent

Gradient descent is a popular optimization strategy that is both powerful and intuitive. It is a first-order iterative method, meaning only gradient information is required, where one attempts to find a local optima by taking steps in a direction proportional to the negative gradient. The procedure is relatively simple: starting at some initial point $\vec{x}^0$ in the domain of the problem, we update our current estimate of the solution $\vec{x}^k$ via the following equation:

$$\vec{x}^{k+1} = \vec{x}^k - \alpha_k \nabla f(\vec{x}^k). \tag{3.25}$$

If f is convex and the gradient is Lipschitz-continuous, convergence to a local optimum can be guaranteed. When f is convex, all local optima are global, and so gradient descent can converge to the global solution.

When f is smooth and the gradient is relatively easy to compute, the crux of the problem may be in choosing the step size α_k at iteration k. For *nice* functions, the value of the step size may be a predetermined quantity for each iteration; however, this is seldom the case in practice. There exist various methods for computing an appropriate step size, most of which rely on some type of line search that satisfies a particular set of conditions. The particular choice in step size may significantly affect the rate of convergence, and in the non-convex case, the quality of the solution obtained. A backtracking line search is the most common approach; this method is simple and tends to work well in practice. In section 3.8.4 we describe the implementation details of the backtracking line search. When feasible, we may be able to compute an exact line search at each iteration, allowing us to maximize our efforts along the direction of the gradient,

$$\alpha^* = \underset{t \geq 0}{\operatorname{argmin}} f(\vec{x} - t\nabla f(\vec{x})).$$

However, this approach may be less computationally efficient and only nominally better in decreasing the objective value when compared to the backtracking line search. Therefore, in most applications the backtracking line search is the preferred method.

Gradient descent algorithms can naturally handle simple constraints such as affine ($A\vec{x} \leq b$) and bound constraints ($\vec{x} \in [l, u]$). Projected gradient descent minimizes a function by moving in the direction of the negative gradient, and then projecting onto the feasible set

$$\begin{aligned} \min_{\vec{x}} \quad & f(\vec{x}) \\ \text{subject to} \quad & \vec{x} \in C \end{aligned} \tag{3.26}$$

The final projected gradient algorithm is

$$\vec{y}^{k+1} = \vec{x}^k - \alpha_k \nabla f(\vec{x}^k)$$

$$\vec{x}^{k+1} = \underset{\vec{x} \in C}{\operatorname{argmin}} \, ||\vec{y}^{k+1} - \vec{x}||$$

Thus, projected gradient descent can be readily applied to bilinear programs with affine constraints, i.e., A_i equal to the zero matrix for $i \in \mathcal{I} \cup \mathcal{E}$.

3.6 Quasi-Newton Methods for Solving QPs

Newton's method is a popular second-order algorithm that benefits from more accuracy and a higher convergence rate than first-order methods. The standard Newton's method is an iterative technique that attempts to find the root of a differentiable function f. However, it may also be applied to optimization programs wherein one attempts to find the root of the derivative of a twice-differentiable function, i.e., a local optimum. This process requires the computation of the inverse Hessian matrix at each iteration to update the current estimate of the solution

$$\vec{x}^{k+1} = \vec{x}^k - H_k^{-1} \nabla f(\vec{x}^k)$$

where $H_k^{-1} = \nabla^2 f(\vec{x}^k)$, the Hessian (second-derivative in univariate calculus) of the function f at the current estimate. Oftentimes, Newton's method is modified to include a small step size $0 < \alpha < 1$ to guarantee sufficient decrease at each iteration

$$\vec{x}^{k+1} = \vec{x}^k - \alpha H_k^{-1} \nabla f(\vec{x}^k).$$

While Newton's method is a second-order method that benefits from rapid convergence (the Hessian greatly speeds up convergence as the algorithm gets closer to the true solution), it is more computationally expensive and only applicable to unconstrained optimization problems. For positive definite quadratic functions, i.e., functions of the form $f(\vec{x}) = \vec{x}^T A\vec{x} + b^T\vec{x}$ such that $A \succ 0$, it is able to obtain the minimizer with just one iteration; however, if the initial point $\vec{x}^0$ is far from the true solution, the Hessian may not be positive definite, and thus the update direction may not be a descent direction.

Nevertheless, there exist a class of algorithms motivated by Newton's method called quasi-Newton methods that use secondary information about the function, in addition to the gradient, without ever having to explicitly compute the Hessian. These methods are sometimes referred to as "pseudo second-order" methods and can handle constraints similar to projected gradient descent, making them applicable to our study. Such methods include the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, its low-memory version the L-BFGS method, and the Barzilai-Borwein method [Fle13, CQ15, BLNZ95, BB88].

Quasi-Newton algorithms approximate the Hessian by implementing a generalization of the secant method using the gradient updates at each iteration. As in the derivation of Newton's method, we define the quadratic approximation of the function f at the current $\vec{x}_k$,

$$f(\vec{x}_k + \vec{s}) \approx q_k(\vec{s}) = f(\vec{x}_k) + \nabla f(\vec{x}_k)^T\vec{s} + \frac{1}{2}\vec{s}^T H_k\vec{s}, \tag{3.27}$$

where $\vec{s} = \vec{x} - \vec{x}^k$. Minimizing the quadratic approximation yields the update $\vec{x}^{k+1} = \vec{x}^k - H_k^{-1}\nabla f(\vec{x}^k)$, with $\vec{s}^{k+1} = \vec{x}^{k+1} - \vec{x}^k = -H_k^{-1}\nabla f(\vec{x}^k)$, the Newton direction. Taking the derivative of the quadratic model at $\vec{x}^{k+1}$ yields

$$\nabla f(\vec{x}) \approx \nabla f(\vec{x}^{k+1}) + H_{k+1}(\vec{x} - \vec{x}^{k+1}). \tag{3.28}$$

By replacing $\vec{x} = \vec{x}^k$ and letting $\vec{s}_k = \vec{x}^{k+1} - \vec{x}^k$ and $\vec{y}_k = \nabla f(\vec{x}^{k+1}) - f(\vec{x}^k)$, we have

$$\vec{y}_k \approx H_{k+1}\vec{s}_k. \tag{3.29}$$

Thus, we can use the gradient information $\vec{y}_k$ to update our current approximation of the Hessian at each iteration. Methods such as BFGS and L-BFGS use symmetric rank-2 updates based on $\vec{s}_k$ and $\vec{y}_k$ to update H_k; however, these methods do not scale well with the dimension of the problem. For problem instances involving millions of variables, storing the matrix updates can be cumbersome. Barzilai and Borwein [BB88] attempt to alleviate this issue by further simplifying the approximate Hessian to a constant-diagonal matrix: $H_k \approx \lambda_k I$, where I is the identity matrix and $\lambda_k > 0$. This "matrix-free" approximation significantly reduces the computational and storage complexity of the problem. Naturally, λ_k is chosen to satisfy the secant approximation of the function at the current iterate

$$\lambda_{k+1} = \frac{\vec{s}_k^T \vec{y}_k}{||\vec{s}_k||^2}, \tag{3.30}$$

or

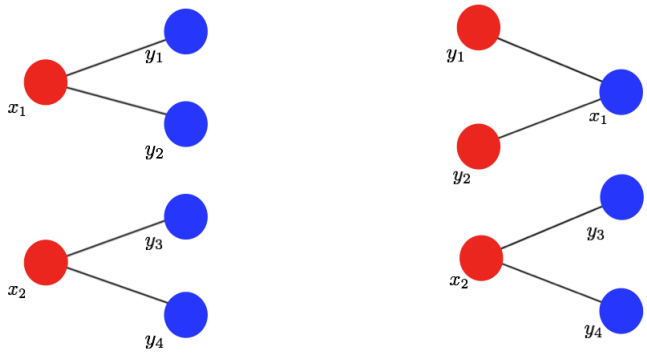
$$\lambda_{k+1} = \frac{||\vec{y}_k||^2}{\vec{s}_k^T \vec{y}_k}. \tag{3.31}$$

3.7 Bipartite Graphs and Bilinear Optimization

The problems we consider for bilinear and quadratic programming are often large with thousands of variables. However, the problems are sparse with each constraint involving just a tiny fraction of these variables. This points out the need for simplification techniques and approaches to solving bilinear problems that exploit this sparsity to make current approaches more efficient. Bipartite graphs offer one such approach by allowing us to divide the set of variables into two disjoint and independent sets, $\mathcal{X}$ and $\mathcal{Y}$, such that edges connecting a variable (or vertex) $x \in \mathcal{X}$ to one in $\mathcal{Y}$ define a bilinear relationship. A bipartite graph that is not connected may have more than one bipartition [CZ08]. Thus, we can *choose* which set of variables to update via some optimization rule, while maintaining the integrity of the original BLP.

Next, we present an example to make clear the connection between bilinear functions and bipartite graphs.

Figure 3.3: Bipartite graph denoting node connectivity between two sets of disjoint nodes. (Left) Nodes x_1 and x_2 belong to set 1 and nodes y_1, y_2, y_3 and y_4 belong to set 2. (Right) Nodes y_1, y_2, and x_2 belong to set 1 and nodes x_1, y_3, and y_4 belong to set 2.



Example Define the bilinear function $f(x, y) = \vec{x}^T B\vec{y}$ with $(x_1\ x_2)^T \in \mathbb{R}^2$, $(y_1\ \ldots\ y_4)^T \in \mathbb{R}^4$ and

$$B = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Clearly, the bilinearity arises from the product terms x_1y_1, x_1y_2, x_2y_3, and x_2y_4. In Figure 3.3 (left) we illustrate the example of an associated bipartite graph where we've indicated the disjoint sets by color. However, we could have analogously defined the function as $\tilde{f}(\tilde{x}, \tilde{y}) = \tilde{x}^T \tilde{B}\tilde{y}$ where $\tilde{x} = (y_1, y_2, x_2)^T \in \mathbb{R}^3$ and $\tilde{y} = (x_1, y_3, y_4)^T \in \mathbb{R}^3$ and

$$\tilde{B} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

whose bipartite graph partition is illustrated in Figure 3.3 (right).

Figure 3.3 has two connected components; that is, two subgraphs in which any two vertices are connected to each other by paths, and there exist no paths connecting vertices from one connected component to the other.

Furthermore, a graph is bipartite if and only if it does not contain an odd cycle and a bipartite graph with n connected components has 2^{n-1} possible configurations. The graphs above contain two connected components and as such, there are two possible configurations for the disjoint sets.

Definition 3.7.1. (Cycle) A cycle is a path of edges and vertices wherein a vertex is reachable from itself. A cycle is called odd if the length of the cycle is odd, and even if the length is even.

We let $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, E)$ denote a bipartite graph whose vertex set has the partition $V = \mathcal{X} \uplus \mathcal{Y}$, with E denoting the edges of the graph.

Claim 3.7.1. Let $\mathcal{G}$ be a bipartite graph, then all cycles in $\mathcal{G}$ have an even number of edges.

Proof. By contradiction, suppose that $\mathcal{G}$ has a cycle with $(2n + 1)$ nodes and edges

$$v_1 \text{ — } v_2 \text{ — } \ldots \text{ — } v_{2n+1} \tag{3.32}$$

Notice that v_{2n+1} must belong to the same partition as v_1 and at the same time, must **not** belong to the same partition. This yields a contradiction. □

An important characterization of bipartite graphs is that a graph is bipartite if and only if it is 2-colorable; that is, the smallest number of colors needed to color the graph $\mathcal{G}$ is less than or equal to two. Thus, we can test whether a graph is bipartite using a coloring-algorithm. This can be done in linear time using a depth-first search, where either a two-coloring or an odd cycle is returned. In the spirit of block-coordinate descent, given a bilinear program whose variable connections form more than one connected component, we can use the connection between BLPs and bipartite graphs to exploit the problem sparsity and flexibility to potentially evade saddle point solutions. The idea being, if a specified solution does not satisfy the necessary conditions to be a local (or global) minimizer, can we gain progress by traversing along another direction, i.e., interchanging the variables over which we optimize? Furthermore, can we improve the algorithm's performance by balancing out the number of variables used for each LP solve?

3.8 Solvers Used for BLP Empirical Comparisons

In a recent publication [GBSBS18], we empirically tested several approaches to solving BLPs, including alternating minimization. The approaches implemented adapt well-known algorithms to the special case of bilinear programs, and use off-the-shelf tools for nonlinear programming. We

considered both local and global nonconvex solvers, and introduced a novel augmented Lagrangian and a novel graph-theoretic approach to solving general BLPs. In this section, we review a set of solvers beyond alternating minimization that are used as a basis of comparison for the problems introduced in sections 3.9 and 3.10. A brief description of each tool is provided. We used our own implementations of alternating minimization and augmented Lagrangian methods as described subsequently. All the solvers are implemented using floating point arithmetic and thus the soundness of the results obtained from these solvers in the context of our benchmark examples is an issue that is not addressed here.

Table 3.1: Solvers used in the numerical experiments.

Solver	Solver Type	Method(s) Employed for Experimental Results
S-Procedure	Local	Semidefinite Programming
Alt Min - GUROBI	Local	Dual-Simplex
Alt Min - SeDuMi	Local	Primal-Dual IP
Bipartite Alt Min - SeDuMi	Local	Primal-Dual IP
Projected Gradient Descent	Local	Non-monotonic BB or `CVX`
Quasi-Newton	Local	Limited-Memory BFGS-Bd Cons
FMINCON	Local	Primal-Dual IP
BONMIN	Global	Hybrid OA/BnC
COUENNE	Global	RLT-based Spatial BnB
IPOPT	Local	Primal-Dual IP
BARON	Global	McCormick Envelope/SDP
Augmented Lagrangian	Local	Gradient-Based/Dual-Simplex

3.8.1 Semidefinite Programming Relaxation

In the case where the bilinear program (3.1) consists of a bilinear objective and a single bilinear inequality constraint, we can transform the problem into a QP and apply the exact SDP relaxation (provided Slater's condition is satisfied). We first compare the effectiveness of this method on a toy problem, for which the true solution is known, against all other methods and solvers to establish the efficacy of each. In general, we are not guarantee boundedness of the objective value over the feasible set. For subsequent examples, bound constraints are placed on the variables to ensure a bounded solution; thus, the SDP formulation will only provide a convex

relaxation for the original problem which will provide a lower bound for the globally optimal value.

3.8.2 Alternating Minimization

The implementation described in section 3.1 used an exact arithmetic LP solver. We examined two efficient floating point LP solvers for alternating minimization. One approach used the dual simplex solver implemented inside the Gurobi tool. Another approach used a floating point primal dual interior point method using the Sedumi solver.

3.8.3 Bipartite Alternating Minimization

For this implementation we begin by solving the given problem using the standard AM routine. The goal of the bipartite reformulation is to improve on the current solution by obtaining a lower objective value. We first build a graph corresponding to the bilinear connections by reformulating the original BLP as a QP and averaging the matrices encoding the bilinear relationships. We then use the `MATLAB` function `conncomp` (part of the graph library) to extract and store the individual connected components into bins. The set is then roughly divided so that the dimension of $\tilde{x}$ is approximately equal to the dimension of $\tilde{y}$. We do this to balance out the effectiveness of each LP solve. However, there may be more than one bipartition corresponding to the potentially high-dimensional problem, thus, we repeat the process of redefining new $\tilde{x}$ and $\tilde{y}$ variables using a random permutation of the connected subgraphs and resolve the problem using the AM procedure, storing each optimal value. The minimum optimal value is compared against the other solvers, and in particular against the standard alternating minimization approach.

3.8.4 Projected Gradient Descent

Since gradient descent can only be performed on a function of one vector variable, we first reformulate each BLP as a QP. This method is used to compare against the other solvers whenever the given problem has bound constraints or affine inequality constraints, but not bilinear. For problems consisting of only bound constraints, projecting onto the set $\vec{x} \in [l, u]$ for finite upper and

lower bounds is trivial. We use `CVX` to compute the new iterate $\vec{x}^{k+1}$ when given a constraint set of affine inequalities. We follow the procedure described in section 3.5 and use a backtracking line search to find a step size satisfying the Wolfe-conditions:

1: **while** $f(\vec{x} - \alpha\nabla f(\vec{x})) > f(\vec{x}) - c\alpha\|\nabla f(\vec{x})\|^2$ **do**

2: $\quad \alpha \leftarrow \rho\alpha$

3: **end while**

with $c = 10^{-4}$ and $\rho = 0.75$ as fixed parameters at each iteration [NW06]. This procedure is used for our randomly generated examples involving both affine inequality and bound constraints.

For the generated examples involving only bound constraints on the variable $\vec{x}$ we implement a Nonmonotone Barzilai-Borwein Gradient Algorithm [XWQ14] that allows for more flexibility in each gradient update. The algorithm uses the Barzilai-Borwein secant approximation to the Hessian at the current iterate to infer the step size in the direction in the negative gradient. However, in the projected gradient descent framework, this update may increase the value of the objective and so a nonmonotone line search is used to control the amount of increase at each iteration. In this case, the descent direction is $\vec{d_k} = -\lambda_k \nabla f(\vec{x}^k)$, in contrast to $\vec{d_k} = -\nabla f(x^k)$ in standard gradient descent, and the nonmonotone line search is used to choose a step size satisfying

$$f(\vec{x}^k - \alpha_k\lambda_k\nabla f(\vec{x}^k)) \leq \max_{0\leq j\leq m(k)} f(\vec{x}^{k-j}) - c\alpha_k\lambda_k\|\nabla f(\vec{x}^k)\|^2,$$

where $m(k)$ is the denotes the memory of the algorithm at the kth iteration. This line search is nonmonotonic in the sense that the new iterate only needs to decrease the objective value by at least the largest objective value for the past $m(k)$ iterations. Clearly, $m(0) = 0$ since there is no memory of previous objective values at the first iteration, and $m(k) = \min\{m(k-1), \tilde{m}\}$ where $\tilde{m}$ is the max-memory specified by the user [XWQ14]. For our comparisons, we chose $\tilde{m} = 5$. To ensure a descent direction, λ_k is chosen to satisfy $\lambda_k = \min\{\lambda_{(max)}, \max\{\lambda_k, \lambda_{(min)}\}\}$ where $\lambda_{(max)} = 10^{20}$ and $\lambda_{(min)} = 10^{-20}$ in our experiments. Thus, if λ_k falls below $\lambda_{(min)}$, the algorithm stagnates since the descent direction d_k is approximately zero and does not change the current iterate.

3.8.5 Quasi-Newton Method

It is important to note that when $\mathcal{I} = \mathcal{E} = \emptyset$, our augmented Lagrangian formulation is an alternative to projected gradient descent. Of the methods mentioned in section 3.6, we chose to employ the L-BFGS-B method, the limited-memory BFGS that allows for bound constraints on the variables. To account for the bound constraints, the method first proceeds as a projected gradient descent to infer which bound constraints are active, and then fixes these variables and optimizes the quadratic program over the remaining free variables. Once the solution $\hat{x}_k$ to the QP is computed, a line search along the direction $d_k = \hat{x}_k - \vec{x}_k$ is performed to find the new iterate $\vec{x}_{k+1}$. The implementation we chose was written in `MATLAB` with a mex wrapper for C. When the constraint set consists only of simple bound constraints on the variables, we use L-BFGS-B to solve our bi-level formulation and project y onto $\vec{l}_y \leq \vec{y} \leq \vec{u}_y$ ourselves. Although this approach is an alternative to projected gradient descent, the two implementations compared here are different. The quasi-Newton approach still "marginalizes" out the y variable each time a new estimate for x is considered. This differs from the projected gradient descent method of section 3.8.4 wherein we first transform the BLP into a QP and then descend in the direction of the negative gradient in the higher-dimensional space. When the constraint set includes more than just simple bound constraints on the variables, we transform all inequality constraints into equality constraints by introducing slack variables and use the augmented Lagrangian framework of section 3.3. In this case, we use L-BFGS-B to solve the outer sub-problem of updating x, and use `CVX` to solve the inner sub-problem of marginalizing out y and s.

3.8.6 FMINCON

`fmincon` is a built-in `MATLAB` nonlinear programming solver [Mata]. It attempts to return feasible stationary points to problems of the form

$$\begin{aligned} \min_{\vec{x}} \quad & f(\vec{x}) \\ \text{subject to} \quad & c(\vec{x}) \leq 0 \\ & c_{\text{eq}}(\vec{x}) = 0 \\ & A_i\vec{x} \leq \vec{b}_i \\ & A_e\vec{x} = \vec{b}_e \\ & \vec{l}_x \leq \vec{x} \leq \vec{u}_x \end{aligned}$$

where both the objective and constraints can be nonlinear functions. The method makes no attempt to find a global optimal point. In order to apply `fmincon` to (3.1), we reformulate the problem into (3.8) as mentioned above. Since the problems we will be comparing are sparse, with a few examples being large scale, we use the solver's primal-dual interior-point algorithm and supply the gradient function. Other supplied algorithms include Trust Region Reflective, Active Set, and Sequential Quadratic Programming; however, Active Set and SQP are not large scale algorithms and the problems considered do not meet the required conditions to apply the Trust Region Reflective algorithm.

3.8.7 BONMIN

`BONMIN` (**B**asic **O**pen-source **N**onlinear **M**ixed **IN**teger programming) is an open source software for solving mixed-integer non-linear programs (MINLPs) of the form

$$\begin{aligned} \min_{\vec{x}} \quad & f(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) \leq 0 \quad i = 1 \dots m \\ & \vec{l}_x \leq \vec{x} \leq \vec{u}_x \\ & x_i \in \mathbb{Z}, \quad i \in I \subseteq \{1, \dots, n\} \\ & x_i \in \mathbb{R}, \quad i \notin I \end{aligned} \tag{3.33}$$

where f and g are assumed to be twice continuously differentiable. The software is distributed by the Computational Infrastructure for Operations Research (COIN-OR). `BONMIN` is equipped with several nonlinear programming algorithms based on branch-and-bound, branch-and-cut, and outer-approximation techniques. The branch-and-bound approach considers polyhedral subsets of the state space and solves a continuous relaxation of the problem containing only a subset of the variables in search for an improvement on the lower bound of the approximation. The branch-and-cut technique uses the same approach to infer an optimal set of candidate solutions but further refines the estimate of the lower bound by implementing nonlinear lift-and-project cutting plane methods to tighten the corresponding relaxation. `BONMIN` uses COIN-ORs non-linear programming-based interior-point solver `IPOPT`, which we introduce below, for its branch-and-bound algorithm. The outer-approximation method is a technique for recasting the original MINLP into a relaxed linear representation by removing any nonlinearities from the objective and adding their linearized variants to the constraint set. It is an iterative procedure that appends newly computed bound constraints found at each iteration to tighten the relaxation. It implements `IPOPT` to solve each NLP and COIN-ORs branch-and-cut algorithm `Cbc` to solve each MILP. The final algorithm offered by `BONMIN` is a hybrid outer-approximation based branch-and-cut algorithm. If either or both f and g are nonconvex then the algorithms are only meant to be heuristics [BBC$^+$08]. Thus there is no guarantee of a global optimal point, but unlike `fmincon` which returns as soon as it has found an approximate stationary point, the `BONMIN` solver attempts to find a global optimal point. For all numerical comparisons in Section 9 we implemented each of `BONMIN`s four solvers and achieved similar results; therefore, the results reported are the results of the hybrid solver.

3.8.8 COUENNE

`COUENNE` (**C**onvex **O**ver and **U**nder **EN**velopes for **N**onlinear **E**stimation) is an open source software primarily developed by Pietro Belotti to solve MINLPs of the form (3.33), where f and g_i are possibly nonconvex [coi16]. The software is also distributed by COIN-OR. `COUENNE` is a RLT-based spatial branch-and-bound algorithm, meaning that the nonconvex terms are replaced

with their convex envelopes and branching can occur on either integer or continuous variables. It ensures global optimal solutions of convex MINLPs and aims to find global optima of nonconvex MINLPS. `COUENNE` is built on top of `BONMIN` and utilizes many of the same options and solvers. More generally, it is a branch and cut algorithm that implements linearization, branching, MINLP heuristics to find feasible solutions, and bound reduction techniques [Bel09]. We emphasize that the software attempts to find a global solution, and thus is naturally slower than methods like `fmincon`.

3.8.9 IPOPT

`IPOPT` (**I**nterior **P**oint **Opt**imizer) is an open source software package for large-scale nonlinear optimization that is distributed by COIN-OR. `IPOPT` supports the same general framework as `COUENNE`; however, it further requires that f and all g_i be sufficiently smooth (at least continuously differentiable). Moreover, `IPOPT` aims to find local solutions to (3.33), like `fmincon`, as opposed to global solutions like `COUENNE`. The `IPOPT` algorithm implements a primal-dual interior-point method that uses line searches based on filter methods. Filter methods offer an alternative to merit functions in that iterates are accepted if they either improve the objective function or improve the constraint violation as opposed to a combination of the two [WB06].

3.8.10 BARON

`BARON` (**B**ranch-**A**nd-**R**educe **O**ptimization **N**avigator) is a global optimization software for solving MINLPs that was developed by Nikolaos Sahinidis at the University of Illinois-Urbana Champaign. It is a branch-and-bound algorithm that has the ability to employ other solvers to solve appropriate subproblems. Given a BLP, `BARON` generates McCormick Envelopes to relax and provide a lower bound for the original problem. Further measures are then taken to improve the accuracy of the computed variables by transforming the relaxed problem into an SDP. `BARON` also employs linear approximations, including outer-approximations, to speed up the computation of the relaxed problem. While this is only a high-level description that is by no means complete, it

provides a brief explanation of the general methods employed by `BARON` for solving BLPs. `BARON` requires that all nonlinear variables and expressions be bounded from above and below to guarantee global optimality. If bounds are not provided, it utilizes appropriate algorithms to infer bounds [Sah17, TS05].

3.9 Experimental Results on Randomly Generated Problems

In this section, we apply the algorithms mentioned in the previous section to a series of benchmark examples. All algorithms were implemented in `MATLAB R2016b`. The COIN-OR source code and binaries provided for `COUENNE` and `IPOPT` have interfaces in `AMPL`, which can be interfaced through `MATLAB`. For the alternating minimization procedure we used the `MATLAB` software `CVX` [GB08, GBY08] with the simplex solver `Gurobi` [Opt17] and the interior-point solver `Sedumi` [Stu99]. For each instance, we report the optimal objective reported by the solver.

Dominant Eigenvector: A Toy Problem.

As a toy example used to test the various approaches, we begin by computing the dominant eigenvector of a positive semidefinite operator, $C \in \mathbb{S}^4_+$. Computing the dominant eigenvector of a positive semidefinite operator can be phrased as the following optimization program:

$$\begin{aligned} \max_{\vec{x}} \quad & \vec{x}^T C \vec{x} \\ \text{subject to} \quad & \|\vec{x}\|_2 \leq 1, \end{aligned} \tag{3.34}$$

or equivalently,

$$\begin{aligned} \min_{\vec{x}} \quad & -\vec{x}^T C \vec{y} \\ \text{subject to} \quad & \|\vec{y}\|_2 \leq 1 \\ & \vec{x} = \vec{y} \end{aligned} \tag{3.35}$$

where $\|\cdot\|_2$ is the ℓ_2-norm of a vector, $\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$. By imposing the additional constraint $\vec{x} = \vec{y}$ we can convert the quadratic objective of (3.34) to the bilinear one in (3.35). While this form does not match the general form of a BLP we can still employ the solvers listed above to get

Table 3.2: Results of the dominant eigenvector program (3.35). We denote step-size by SS.

Method	Status	Normalized Residual
SDP	Solved	1.327e-14
Proj Grad (Wolfe SS)	Solved	3.932e-08
Proj Grad (Fixed SS)	Solved	1.592e-05
FMINCON	Solved	9.750e-09
BONMIN	Solved	4.999e-09
COUENNE	Solved	4.000e-03
IPOPT	Solved	3.156e-10
BARON	Solved	2.270e-05
Aug Lag	Solved	5.100e-05

an idea of the expected solver accuracy. For the augmented Lagrangian formulation we remove the slack variable s and replace the LP solve via `CVX` with the closed-form projection

$$\vec{y}_x = \begin{cases} \hat{x}, & \text{for } \|\hat{x}\|_2 \leq 1 \\ \frac{\hat{x}}{\|\hat{x}\|_2}, & \text{otherwise} \end{cases} \tag{3.36}$$

where $\hat{x} = -\frac{1}{\mu}(C\vec{x}^k + \lambda^k) + \vec{x}^k$. Table 3.2 summarizes the performance of various solvers in terms of the final solution status and the normalized residuals. While this toy example gives us an idea for each solver's expected accuracy, it is also meant to test the implementation of each method. That is, we see that all algorithms pass this basic sanity check. The table does not include alternating minimization, since every feasible point is also a saddle point for this problem.

Bilinear Programs of Varying Complexity

We consider various cases of bilinear programs where we maintain the bilinear objective function but vary the set of constraints. For each problem case, we vary the sparsity structure of the bilinear terms by increasing the density of non-zero entries using the percentages 0.05%, 0.5%, 1%, 30%, 60%, and 100% for comparison. In the first case we also approximate the scalability of each solver by measuring the runtime for problem instances of increasing dimensions. Each program is randomly generated using a seed to ensure reproducibility. The first example considered contains only simple bound constraints (i.e., $\mathcal{I} = \mathcal{E} = \emptyset$); therefore, we do not use

the full augmented Lagrangian expression, but the quasi-Newton algorithm `L-BFGS-B` to solve the bi-level formulation. Furthermore, due to the algorithmic similarities between `BARON` and `COUENNE`, and `BARON`s poor scalability, we do not include it in our set of solvers used to solve the randomized examples. For compactness, we abbreviate the solvers used in the plots. Each abbreviation is displayed in Table 3.3.

Table 3.3: List of solver abbreviations for subsequent figures.

Abbreviation	Method
SDP	Semidefinite Programming Relaxation (S-lemma)
PGD	Projected Gradient Descent (Line search specified in subsections)
Alt Min (or AM)	Alternating Minimization
BAM	Bipartite-Graph Inspired Alternating Minimization
Q-N	Quasi-Newton Method (`L-BFGS-B`)
Aug Lag	Augmented Lagrangian
fmincon (or fmin)	`fmincon`
IPOPT	`IPOPT`
COUENNE	`COUENNE`

3.9.1 Case 1: Simple Bound Constraints

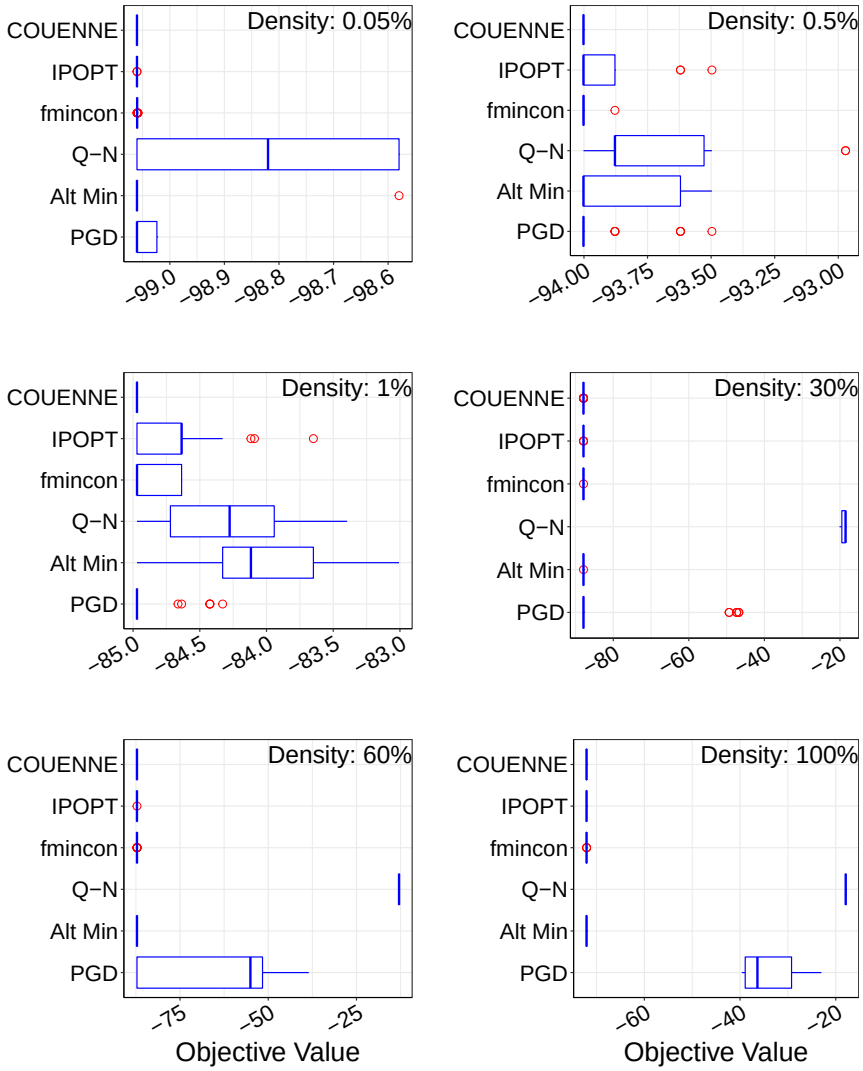
We first consider the simplest case, where the constraint set is composed of bound constraints on x and y.

$$\begin{aligned} \min_{\vec{x},\vec{y}} \quad & \vec{x}^T A_0 \vec{y} + \vec{b}_0^T \vec{x} + \vec{c}_0^T \vec{y} \\ \text{subject to} \quad & \vec{l}_x \leq \vec{x} \leq \vec{u}_x,\ \vec{l}_y \leq \vec{y} \leq \vec{u}_y \end{aligned} \tag{3.37}$$

In order to guarantee a bounded objective value, we restrict both $\vec{x}$ and $\vec{y}$ to the compact set $S = \{(\vec{x}, \vec{y}) \mid 0 \leq x_i \leq 1,\ 0 \leq y_i \leq 1\}$. Generating a feasible initial point is trivial, as we can just assign values to $\vec{x}^0$ and $\vec{y}^0$ by drawing from a uniform distribution over the interval $[0, 1]$. In the first set of comparisons, we measure the median objective values over 30 different initializations and display their variability in Figure 3.4. For comparing the efficacy of each solver as a function of sparsity, we fix the dimension of each variable, in this case $\vec{x} \in \mathbb{R}^{40}$ and $\vec{y} \in \mathbb{R}^{200}$.

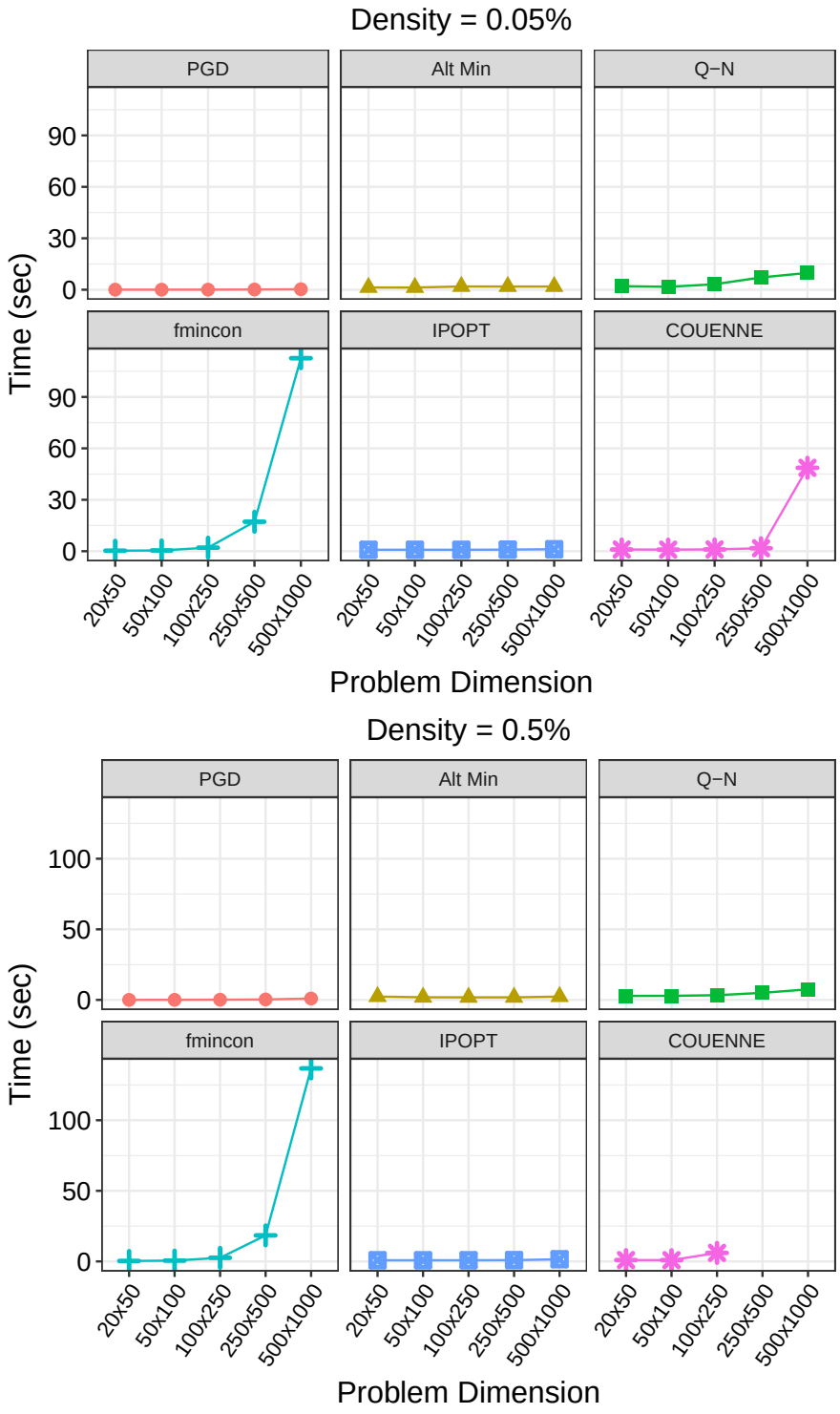
Since `COUENNE` is a global solver, its objective value represents the global minimum of our

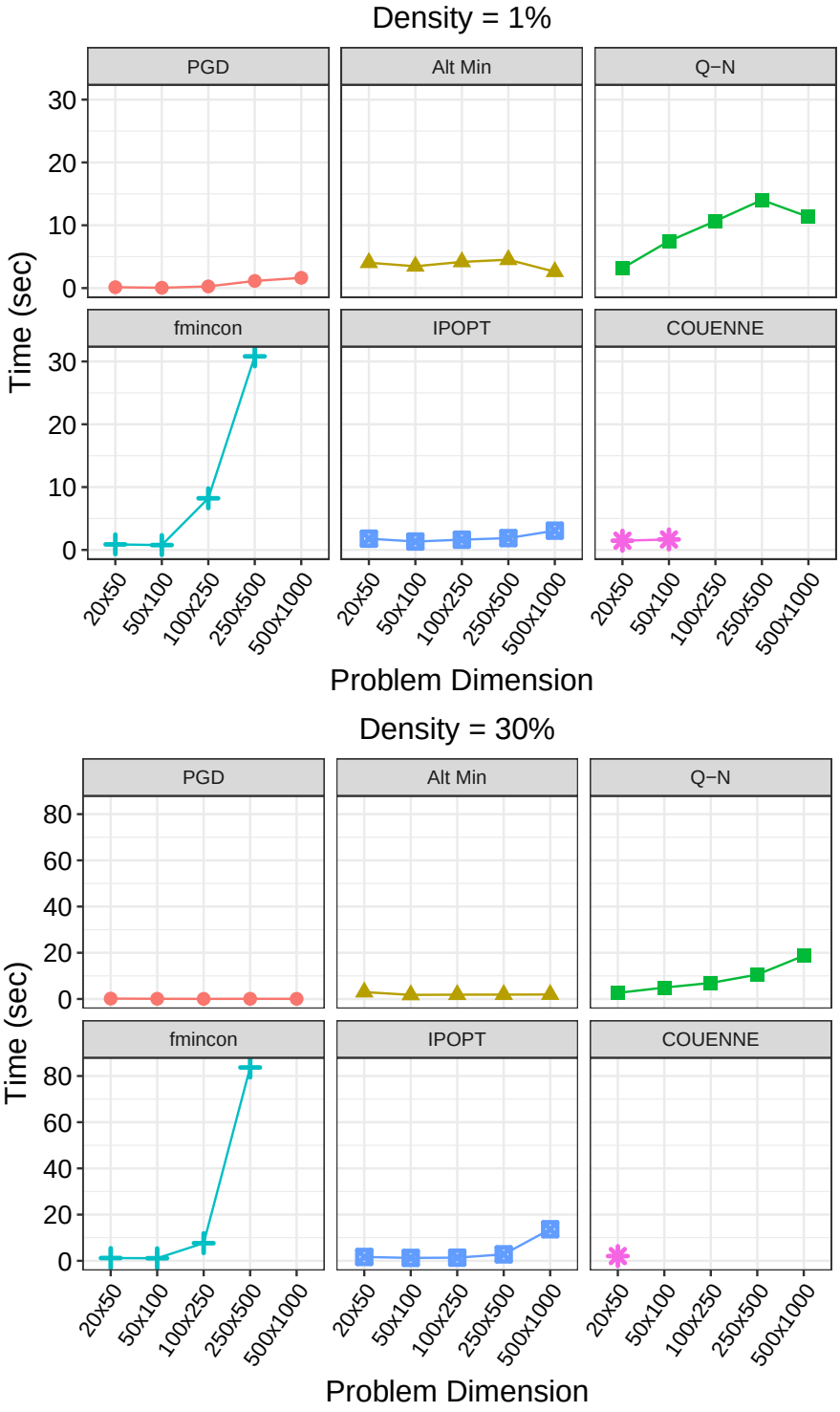
Figure 3.4: Experimental results displaying median objective value and variation over 30 different initializations. The objective values are displayed on the x-axis and the solver on the y-axis. Red circles denote outlier objective values. In order, the solvers displayed are: `COUENNE`, `IPOPT`, `fmincon`, a Quasi-Newton approach (`L-BFGS-B`), Alternating Minimization, and a Projected Gradient Descent approach with a nonmonotonic Barzilai-Borwein line search.



problem instance, and it can be used as a certificate for each solver's ability to obtain the global minimum. For a fixed problem size, as the number of bilinear connections increase we notice that the Quasi-Newton and projected gradient descent approaches become more sensitive to the starting point $\vec{x}^0$. Their ability to reach the global optimum degrades as the complexity of the problem increases. Furthermore, we note that alternating minimization is competitive with solvers such as

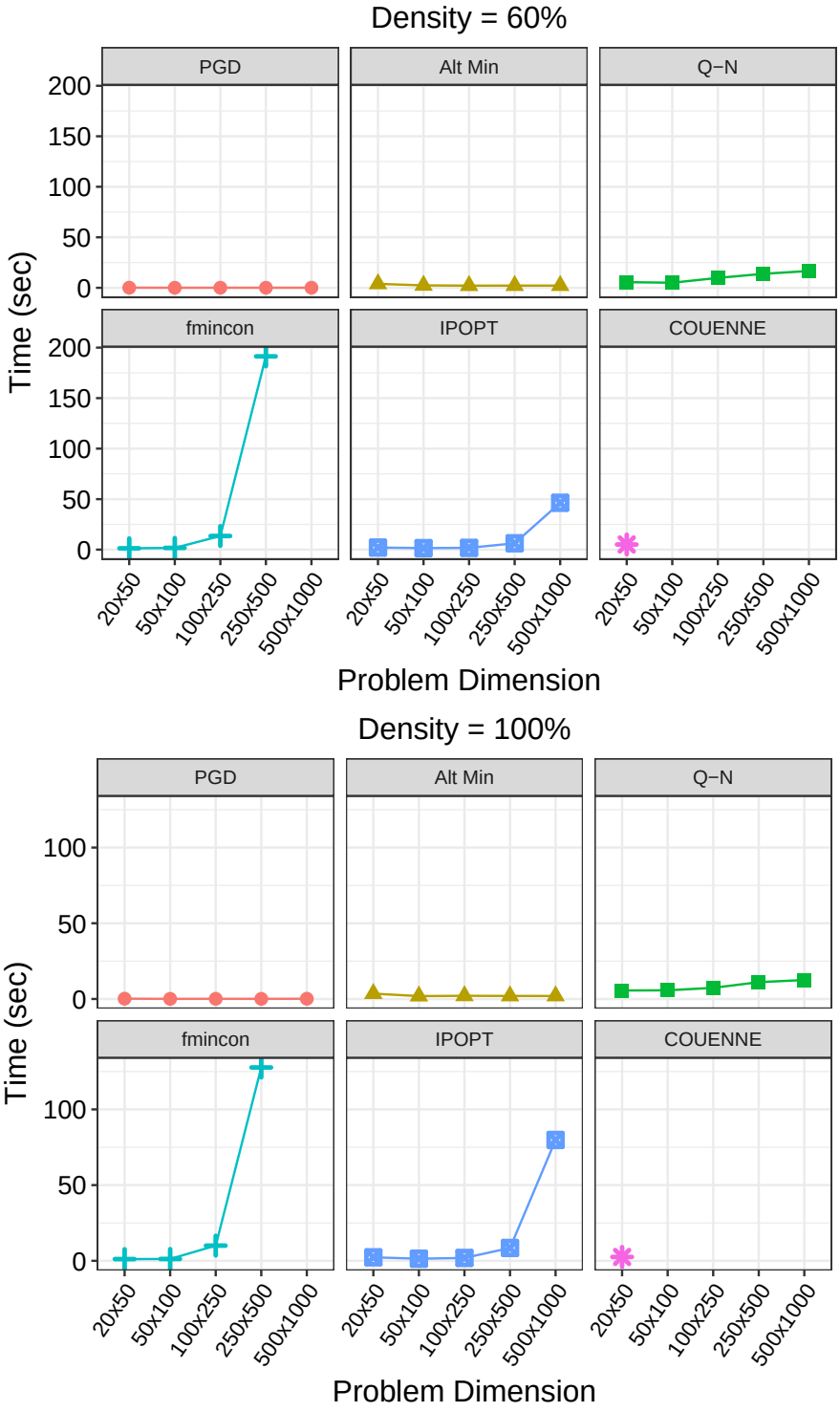
Figure 3.5: Runtime comparison (in seconds) for bound-constrained bilinear programs (3.37) of varying dimension. The plots are distinguished by the density of non-zero terms in A_0.





COUENNE and IPOPT in finding the global minimum. Although COUENNE is a global solver, it does not scale well with the complexity of the problem as we show in the runtime experiments below.

In this case, the bipartite-graph formulation of alternating minimization discussed in section



3.7 will behave similarly to the standard AM method. For a fixed $\vec{x}_*$, (3.37) is a LP in $\vec{y}$ and be can written as

$$\begin{aligned} \min_{\vec{y}} \quad & \tilde{c}^T \vec{y} \\ \text{subject to} \quad & 0 \leq \vec{y} \leq 1 \end{aligned} \tag{3.38}$$

where $\tilde{c} = A_0^T \vec{x}_* + \vec{c}_0$. The solution to this program is

$$\vec{y}_* = \begin{cases} 1 & \text{if } \tilde{c}_i < 0 \\ 0 & \text{if } \tilde{c}_i \geq 0, \end{cases} \tag{3.39}$$

independent of how we partition the variable set $(\vec{x}, \vec{y})$. The same applies to holding $\vec{y}_*$ fixed and solving a LP over the variable $\vec{x}$. The partitioning could, however, alter the sign of $\vec{c}$, thus changing objective value. However, since both methods force the values of x and y onto the boundary of the feasible region, there is no reason to believe the bipartite formulation would offer us much of a benefit over the standard procedure. After one iteration the program returns a solution on the boundary of the compact set. If we aren't restricted to the compact set S, then the program is unbounded.

For runtime comparisons, we time each solver using `MATLAB`'s `tic/toc` function for bilinear problems of increasing dimension and density. The results are displayed in Figure 3.5. Each solver is timed out after ten minutes, and thus their point-values are absent from their respective plots. `IPOPT`, alternating minimization, `L-BFGS-B`, and the projected gradient descent approach scale remarkably well with the dimension of the problem, while `fmincon` and `COUENNE` do not. As can be seen from the experimental results, as the complexity (size and density) of the problem increases, the amount of time required to compute a global solution increases drastically for `COUENNE`. When the bilinear objective term is fully dense, it can efficiently solve *small* problems; the runtime of the solver is exponential in the number of variables that branching occurs on. In the following section, we test a suite of solvers on the Template Abstract Domain benchmark problems and find that `COUENNE` is unable to terminate within a fixed amount of time due to the complexity of the examples. Furthermore, we note that `fmincon`'s runtime increases exponentially with the problem size and density; however, it is able to handle a larger range of problems compared to `COUENNE`.

3.9.2 Case 2: Affine Inequality Constraints

Next, we consider the case where the objective is a bilinear function and our constraint set consists of a set of affine inequality and simple bound constraints on $\vec{x}$ and $\vec{y}$

$$\begin{aligned} \min_{\vec{x},\vec{y}} \quad & \vec{x}^T A_0 \vec{y} + \vec{b}_0^T \vec{x} + \vec{c}_0^T \vec{y} \\ \text{subject to} \quad & A\vec{x} + B\vec{y} \leq \vec{r}, \\ & \vec{l}_x \leq \vec{x} \leq \vec{u}_x,\ \vec{l}_y \leq \vec{y} \leq \vec{u}_y \end{aligned} \tag{3.40}$$

We again restrict $\vec{x}$ and $\vec{y}$ to the compact set $S = \{(\vec{x}, \vec{y}) \mid 0 \leq x_i \leq 1,\ 0 \leq y_i \leq 1\}$ to ensure a bounded solution. The problem instances (3.40) are generated such that we know of at least one feasible point that may not be globally optimal. In defining the affine inequality constraints, we generate multiple feasible points $(\vec{x}_i, \vec{y}_i) \in S$, compute the corresponding $A\vec{x}_i + B\vec{x}_i = \vec{r}_i$ and set $\vec{r} = \max_{0 \leq j \leq |\mathcal{I}|} r_{ij}$. We then choose one of the feasible points used to generate the problem as the initial $(\vec{x}^0, \vec{y}^0)$ for each solver. The objective value for a fixed problem size of varying density is displayed in Figure 3.6. Each problem instance is again restricted to $\vec{x} \in \mathbb{R}^{40}$ and $\vec{y} \in \mathbb{R}^{200}$. In general, most solvers do relatively well, and are comparable for problem instances with a relatively low number of bilinear connections. As the density of non-zero terms of A_0 increases, we note that the first solver to perform poorly is our novel augmented Lagrangian approach, followed by alternating minimization and our novel bipartite-graph formulation of alternating minimization.

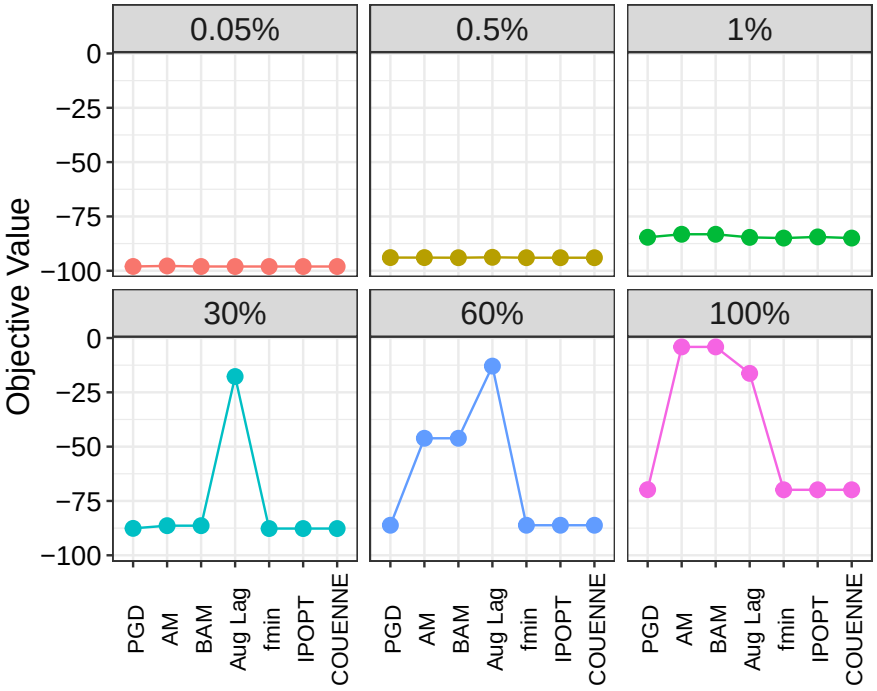
The projected gradient descent approach implemented for this case uses the standard Wolfe conditions to deduce an appropriate step size (**??**). Its ability to obtain the optimal objective value is competitive with the global solver, `COUENNE`, as well as the local solvers, `IPOPT` and `fmincon`, for problem instances of increasing bilinear complexity. Recall the variable update for projected gradient descent:

$$\vec{y}^{k+1} = \vec{x}^k - \alpha_k \nabla f(\vec{x}^k)$$

$$\vec{x}^{k+1} = \operatorname*{argmin}_{\vec{x} \in C} \|\vec{y}^{k+1} - \vec{x}\|$$

The main crux of projected gradient descent in this case is the projection of the new point $\vec{y}^{k+1}$

Figure 3.6: Objective value of (3.40) for various solvers. Each subplot corresponds to a different density of non-zero terms of A_0.



onto the set $C = \{(\vec{x}, \vec{y}) \mid A\vec{x} + B\vec{y} \leq \vec{r},\ 0 \leq x_i \leq 1,\ 0 \leq y_i \leq 1\}$. For our implementation, we used `CVX` interfaced through `MATLAB` to solve each projection.

3.9.3 Case 3: A Single Bilinear Inequality Constraint

In this subsection, we examine the case where our constraint set consists of a single bilinear inequality constraint, as well as bound constraints on $\vec{x}$ and $\vec{y}$

$$\begin{aligned} \min_{\vec{x},\vec{y}} \quad & \vec{x}^T A_0 \vec{y} + \vec{b}_0^T \vec{x} + \vec{c}_0^T \vec{y} \\ \text{subject to} \quad & \vec{x}^T A_1 \vec{y} + \vec{b}_1^T \vec{x} + \vec{c}_1^T \vec{y} \leq r, \\ & \vec{l}_x \leq \vec{x} \leq \vec{u}_x,\ \vec{l}_y \leq \vec{y} \leq \vec{u}_y \end{aligned} \tag{3.41}$$

We define the bilinear constraint by first generating several feasible points $(\vec{x}_i, \vec{y}_i) \in S$ and taking $\vec{r} = \max_i\{\vec{x}_i^T A_1 \vec{y}_i + \vec{b}_1^T \vec{x}_i + \vec{c}_1^T \vec{y}_i\}$, thus guaranteeing a non-empty feasible region. For this com-

parison, we include the semidefinite programming relaxation motivated by the S-lemma discussed in section 3.4. However, by including the set of constraints $\vec{x} \in [\vec{l}_x, \vec{u}_x]$ and $\vec{y} \in [\vec{l}_y, \vec{u}_y]$, we are no longer able to apply the S-lemma [Yak71] and guarantee an exact relaxation. Therefore, we expect our SDP relaxation to yield a loose lower bound on the objective value. Figure 3.7 displays the objective value obtained by each solver. Each subplot represents a different density of non-zero terms in both A_0 and A_1, and the SDP relaxation is included to show the duality gap, $p^* - d^*$, between the optimal value and the lower bound provided by the relaxation.

Figure 3.7: Objective value obtained by various solvers in solving an instance of equation (3.41). The SDP relaxation is included to show the duality gap for varying levels of bilinear connectivity.

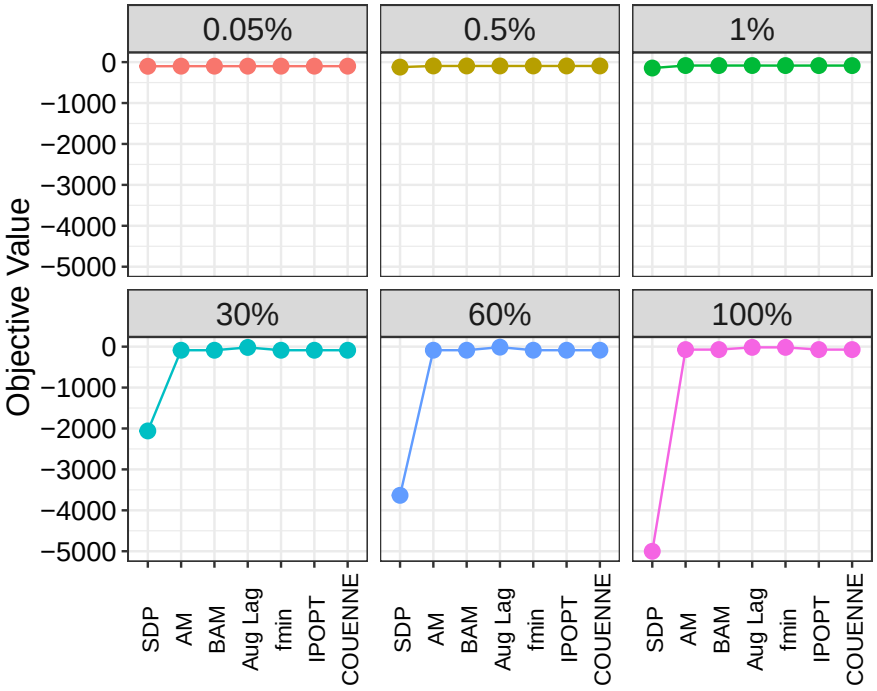
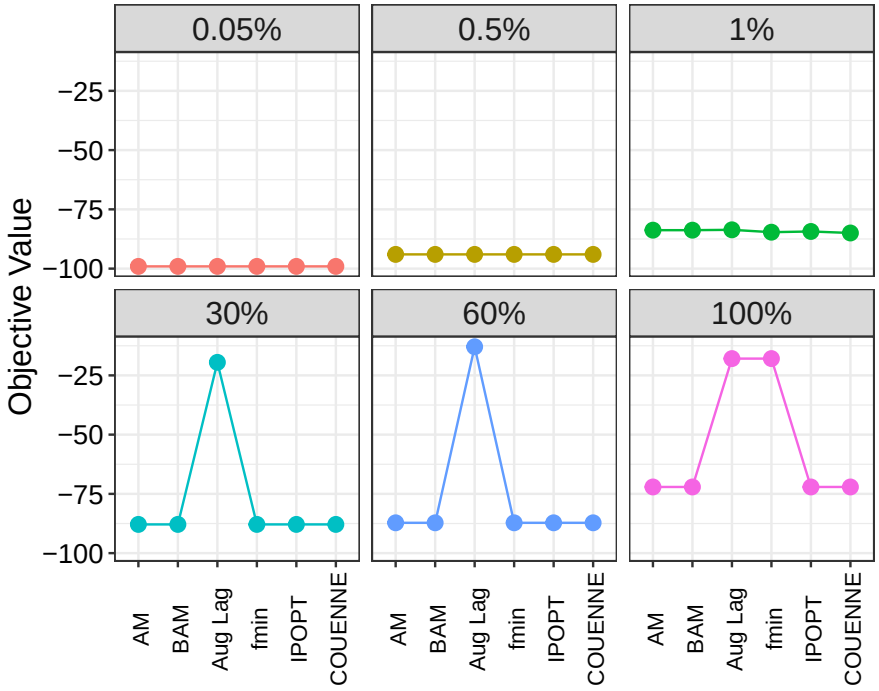


Figure 3.8 displays the same results, but excludes the SDP relaxation in order to better judge the performance of the other solvers. Once more, we note that as the sparsity level of each bilinear term decreases, the augmented Lagrangian method is more likely to stagnate at a non-optimal solution. We observe the same issue with `fmincon` when the matrices A_0 and A_1 are fully dense.

Projected gradient descent is not included in the above experiments since it is unclear how

Figure 3.8: Figure 3.7 without the SDP relaxation to show differences in the remaining solvers.



one would project onto the bilinear inequality constraint set.

3.10 Template Abstract Domain Experimental Results

In this section, we apply the algorithms mentioned in section 3.8 to a series of benchmarks that include Example 2.5.3 discussed previously and benchmarks 2-5 in Table 2.1. Benchmarks 1 and 6-9 are not included in the numerical comparisons since the algorithms implemented do not scale well with the size of the problems. For each instance, we report the optimal objective reported by the solver along with the **feasibility gaps** that are given by the largest absolute violation of equality constraints and largest inequality constraint violation to deduce feasibility of the resulting solution.

Method	Obj Value	$\max\{\lvert c_{\text{eq}}(x)\rvert\}$	$\max\{c_{\text{ineq}}(x)\}$	Status
Alt Min Gurobi	**1.000e+00**	**0.000e+00**	**0.000e+00**	Solved
Alt Min Sedumi	1.970e+02	**0.000e+00**	1.421e-14	Solved
FMINCON	**1.000e+00**	**0.000e+00**	**-1.603e-08**	Solved
BONMIN	**1.000e+00**	2.675e-08	1.020e-06	Solved
COUENNE	**1.000e+00**	8.963e-09	5.108e-07	Solved
IPOPT	**1.000e+00**	1.749e-08	1.011e-06	Solved
BARON	**1.000e+00**	5.845e-09	7.437e-09	Solved
Aug Lag	2.75e+02	9.925e-07	**-8.074e-08**	Timed Out

Table 3.4: Results on Example 2.5.3, m=2, n=6, NumCons=4.

Example Table 3.4 applies these approaches to the relatively small problem obtained from the program shown in Example 2.5.3. The objective in this example is set to $c_1 - c_2$, wherein the invariant that we seek has the form $x_1 \leq c_1 \ \wedge \ x_1 \geq c_2$. Starting with the initial solution $c_1 = 100, c_2 = 0$, alternating minimization using an interior point solver and 5 other approaches compute the best solution $c_1 - c_2 = 1$.

Tables 3.4, 3.5, 3.6, 3.7, and 3.8 compare the performance of various algorithms over benchmark IDs 2-5 from Table 2.1. In each table, the highlighted entries indicate "desirable results" in terms of a negative solution and corresponding feasibility gaps of 10^{-7} or less. Due to the different types of solvers and the use of Matlab interfaces, we will not compare the time taken by each approach. The key conclusion is that alternating minimization approach is by far the best

in terms of solving these benchmarks with acceptable feasibility gaps. This is surprising given the propensity of this approach to get "stuck" in a saddle point. Furthermore, floating point alternating minimization approaches achieve acceptable feasibility gaps and the approach itself lends to exact arithmetic implementation as shown in Section 2.6.

Method	Obj Value	$\max\{\lvert c_{\text{eq}}(x)\rvert\}$	$\max\{c_{\text{ineq}}(x)\}$	Status
Alt Min Gurobi	**-3.175e-02**	**3.980e-11**	**4.818e-14**	Solved
Alt Min Sedumi	**-9.344e-01**	1.632e+01	7.915e-01	Infeasible
FMINCON	-7.064e+06	7.502e-02	2.984e-01	Timed Out
BONMIN	-1.022e+21	NA	NA	Failed
COUENNE	NA	NA	NA	Timed Out
IPOPT	-3.265e+12	1.962e+08	5.258e+07	Timed Out
BARON	-2.375e+07	4.475e+09	4.096e+13	Timed Out
Aug Lag	3.950e+02	1.140e-01	1.432e-01	Stagnant

Table 3.5: Benchmark # 2 from Table 2.1, m=48, n=264, p=353, NumCons=123.

Method	Obj Value	$\max\{\lvert c_{\text{eq}}(x)\rvert\}$	$\max\{c_{\text{ineq}}(x)\}$	Status
Alt Min Gurobi	**-2.066e-01**	**1.137e-13**	**4.441e-16**	Solved
Alt Min Sedumi	**-2.286e-01**	**2.220e-09**	**5.754e-09**	Solved
FMINCON	8.386e+01	**2.536e-11**	**-2.560e-05**	Timed Out
BONMIN	-6.298e+08	NA	NA	Failed
COUENNE	NA	NA	NA	Timed Out
IPOPT	-3.805e+08	1.087e+08	1.547e+11	Timed Out
BARON	-1.102e+13	3.148e+12	1.786e+11	Timed Out
Aug Lag	3.310e+00	1.029e-05	5.331e-06	Stagnant

Table 3.6: Benchmark # 3 from Table 2.1, m=24, n=72, p=161, NumCons=51.

Method	Obj Value	$\max\{\lvert c_{\text{eq}}(x)\rvert\}$	$\max\{c_{\text{ineq}}(x)\}$	Status
Alt Min Gurobi	**-1.000e-01**	**1.138e-14**	1.213e-13	Solved
Alt Min Sedumi	**-1.000e-01**	**2.064e-13**	**-5.174e-14**	Solved
FMINCON	1.208e+00	2.211e-02	7.211e-02	Infeasible
BONMIN	-5.641e+09	NA	NA	Failed
COUENNE	NA	NA	NA	Timed Out
IPOPT	3.741e+10	2.799e+10	5.144e+12	Timed Out
BARON	-7.834e+13	1.454e+11	7.509e+15	Infeasible
Aug Lag	2.000e+00	7.380e-07	5.306e-07	Solved

Table 3.7: Benchmark # 4 from Table 2.1, m=12, n=20, p=33, NumCons=27.

Method	Obj Value	$\max\{\lvert c_{\text{eq}}(x)\rvert\}$	$\max\{c_{\text{ineq}}(x)\}$	Status
Alt Min Gurobi	**-2.417e-01**	**5.994e-13**	**1.742e-11**	Solved
Alt Min Sedumi	**-2.417e-01**	**8.652e-10**	**2.261e-09**	Solved
FMINCON	2.546e+02	**3.553e-15**	**-9.184e-04**	Infeasible
BONMIN	-6.687e+17	NA	NA	Failed
COUENNE	NA	NA	NA	Timed Out
IPOPT	-1.764e+13	5.408e+09	2.230e+10	Timed Out
BARON	1e+51	NA	NA	Failed
Aug Lag	1.847e+01	1.859e-05	1.198e-06	Stagnant

Table 3.8: Benchmark # 5 from Table 2.1, m=40, n=410, p=681, NumCons=204.

3.11 Discussion

To conclude this topic, we exploited the connection between template domains and bilinear constraints. In doing so, we show that policy iteration (alternating minimization) allows the template directions to be updated on the fly in a property directed fashion. We present preliminary evidence that such an approach can be effective, though many challenges remain.

Furthermore, we introduced two novel approaches to solving bilinear programs: one utilizing the augmented Lagrangian framework and the other adapting concepts from graph-theory to the standard alternating minimization approach. We generated a set of test examples to better understand the efficacy of some of the solvers described in section 3.8 on bilinear programs. While it is more desirable to obtain a global solution, methods such as `COUENNE` and `BARON` do not scale well with the dimension of the problem; therefore, their applicability may be limited as demonstrated

with the template domain benchmark experiments. Since branch and bound methods search the entire state space for a global solution, the number of operations required to solve the problem is exponential in the number of variables. Many local solvers perform just as well and can solve the problem much more quickly. Projected gradient descent and our novel augmented Lagrangian method are efficient, but provide unsatisfactory solutions as the density of the bilinear connections grow. We suspect that as the feasible region becomes more *irregular* by the increasing amount of bilinear connections or addition of more complex constraints, these methods begin to suffer. All of the solvers we considered performed well on our toy example (the eigenvalue problem); therefore, we surmise that the constraint set significantly impacts the performance of the algorithm. All things considered, alternating minimization appears to be the most beneficial approach to solving bilinear programs. They scale very well with the size of the problem and produce competitive results, even when the graph is fully connected.

Chapter 4

Information Theoretic Concepts and Applications to Super-Resolution Imaging Techniques

In this chapter we examine how to establish claims of super-resolution; other than just judging by eye, there are not many formal tests. One such quantitative test is Fourier ring correlation, a heuristic technique that compares the spectral information of two images to deduce their likeness, but this test can be easily fooled by algorithms that produce precise (but not accurate) estimates. We propose an alternate method for quantifying super-resolution: we would like to apply information-theoretic concepts to recovering statistics on the image-generation process, before reconstruction. That is, we would like to circumvent the algorithmic reconstruction process altogether and avoid the inclusion of prior knowledge to aid in super-resolution reconstructions. At their most fundamental levels, the quantities of information theory are defined as functionals of probability distributions, and for certain distributions, the maximum amount of information that can be acquired by a process is known in closed form. We introduce the concept of empirically comparing the relative information between two distributions that are relevant to the imaging problem generated by super-resolution techniques for varying signal-to-noise ratios, and to compare those values against the theoretical upper bound, the channel capacity. In what follows, we compare the formulation of our proposed approach of estimating the mutual information between an object and its raw measurements using Monte Carlo simulations and current discretization approaches to estimating mutual information. We discuss some of the inherent difficulties in estimating the entropy and mutual information functions. Ultimately, our aim is to introduce a novel approach

for quantifying the effectiveness of certain super-resolution imaging techniques and disentangle physical mechanisms for super-resolution from post-processing mechanisms. By isolating the physical measurement and the noise model, we can study the amount of information contained in raw measurements, independent of post-processing.

4.1 Overview

A conventional microscope can only resolve image details with a line spacing larger than the diffraction limit of the objective lens, d_0. Equivalently, it can only detect the information contained within a circular region of radius $1/d_0$ in Fourier space. Therefore, conventional methods to imaging and phase retrieval can only accurately reconstruct a signal, or image, up to the diffraction limit. In our investigation of phase retrieval algorithms, we found that current approaches rely heavily on additional side information in conjunction with the estimated intensity measurements. This information may be *a priori* knowledge from the application domain; however, there are instances where additional information may be insufficient, or unavailable. Hence, an accurate reconstruction of the original image may be infeasible. Super-resolution imaging is a class of techniques that aims to reconstruct an image that appears to have a higher resolution than allowed by the simple use of diffraction-limited optics. In essence, multiple snapshots of the same image are taken using a structured pattern, or in a localized manner, in hopes of obtaining more information about the imaged sample. While these approaches show promise, there is an associated limit on the amount of information gained from each additional snapshot of the sample, and a limit on the total amount of information gained by implementing various super-resolution techniques.

In what follows, we give a general overview of the phase retrieval (PR) problem and current approaches to estimating the phase information of a sample given only its intensity measurements before moving on to super-resolution imaging. We discuss the PR problem's connection to quadratic programs, along with several approaches (both convex and nonconvex) to solving the problem. In section 4.3 we review a bellwether algorithm that performs remarkably well on the PR problem, given its nonconvex formulation. Beginning with section 4.4, we review the topic of super-resolution

imaging and provide a motivating example for requiring better standards for assessing the effectiveness of a technique. We then discuss the concept of mutual information, and in section 4.5.2 we briefly review current discretization approaches to estimating these quantities before introducing our novel proposal of implementing Monte Carlo methods to estimate the mutual information and use this quantity as a measure for the amount of information gained from a super-resolution technique in section 4.5.3.

4.2 Phase Retrieval as a Specific Case of Quadratic Programming

Consider the complex signal $F(u)$, with modulus $|F(u)|$, and phase $\phi(u)$. The objective of phase retrieval is to algorithmically determine the phase information of a complex signal from its modulus measurements and additional *a priori* information. We define

$$F(u) = |F(u)| \exp(i\phi(u)) = \int_{-\infty}^{\infty} f(x) \exp(-2\pi i x \cdot u) dx, \tag{4.1}$$

where x is an M-dimensional spatial coordinate and u is an M-dimensional spatial frequency coordinate. We are particularly interested in recovering a 2D image X when given its modulus data, $|F(X)|$; that is, the case for which $M = 2$. Knowing the phase information allows us to perfectly reconstruct an image from its Fourier modulus information by applying the inverse Fourier transform.

Phase retrieval is well-known to be an ill-posed problem. At best, we may be able to reconstruct an image up to some global phase ambiguity since

$$|F(cu)| = |cF(u)| = |F(u)| \tag{4.2}$$

where $c \in \mathbb{C}$ has modulus 1. Therefore, it is not possible to distinguish u from cu and the aim is to recover u up to multiplication by a unitary complex number. Furthermore, for an $n \times n$ image, we have n^2 degrees of freedom, without any *a priori* information. Thus, additional constraints, or domain-specific knowledge must be provided for there to be any hope of reconstructing the original

image.

In phase retrieval, we are interested in solving quadratic equations of the form

$$Y = |F(X)|^2 \tag{4.3}$$

where $X \in \mathbb{R}^{n \times m}$ is a real, not necessarily square, image and Y are the intensity, or squared modulus, measurements. Mathematically this can be stated as

$$\begin{aligned} &\text{find} && X \\ &\text{subject to} && Y_{i,j} = |F(X)|^2_{i,j} \quad \text{for} \;\; 1 \leq i \leq n, 1 \leq j \leq m \end{aligned} \tag{4.4}$$

Let $\ell(x, y)$ be a loss function measuring the misfit between both its scalar arguments. We can consider the generalized phase retrieval problem

$$\begin{aligned} &\min_X && \sum_i \sum_j \ell(Y_{i,j}, |F(X)|^2_{i,j}) \\ &\text{subject to} && X \in \mathcal{D} \end{aligned} \tag{4.5}$$

where $\mathcal{D}$ is the image domain, $i \in \{1, \ldots, n\}$, and $j \in \{1, \ldots, m\}$. If we have information about the statistics of our noise, we can use it in our optimization formulation via the maximum likelihood estimation framework. If we assume that our measured images suffer only from white Gaussian noise, then the probability of capturing the measured intensity $Y_{i,j}$ at pixel index (i, j), given the estimate $|F(\hat{X})|^2_{i,j}$, can be expressed as

$$p(Y_{i,j} \mid |F(\hat{X})|^2_{i,j}) = \frac{1}{\sqrt{2\pi N}} \exp\left(\frac{-(Y_{i,j} - \hat{Y}_{i,j})^2}{2N}\right) \tag{4.6}$$

where N is the variance of the white Gaussian noise and $\hat{Y}_{i,j} = |F(\hat{X})|^2_{i,j}$. Assuming that each pixel measurement is independent, the likelihood function can be calculated as $\prod_i \prod_j p(Y_{i,j} \mid |F(\hat{X})|^2_{i,j})$. The goal in maximum likelihood estimation is to maximize the likelihood function; however, due to numerical instabilities and floating-point errors that arise from multiplying many very small numbers together, it is easier to solve this problem by transforming the likelihood function into

the negative log-likelihood and minimizing the resulting function. The monotonicity of the log function preserves all optima, replaces products with summations, and makes the differences in the measured versus estimated data more pronounced. If we assume that the noise-variance, N, is constant and independent of the pixel-intensity measurements, then we arrive at the familiar least-squares optimization problem

$$\begin{aligned} \min_{X,\hat{Y}} \quad & \sum_i \sum_j (Y_{i,j} - \hat{Y}_{i,j})^2 \\ \text{subject to} \quad & \hat{Y}_{i,j} = |F(X)|^2_{i,j}, \quad \forall i,j \\ & X \in \mathcal{D} \end{aligned} \tag{4.7}$$

We specify $\mathcal{D}$ using known properties of the variable. In most imaging applications, one imposes the constraint that each pixel value be real and non-negative, or $\mathcal{D}_0 = \{x \mid x \in \mathbb{R} \text{ and } x \geq 0\}$, with $D = D_0 \otimes D_0 \otimes \ldots \otimes D_0$; in general, the specification of $\mathcal{D}$ is application-specific. The above problem is nonconvex in its constraints, so there may be many stationary points associated with the problem. Furthermore, convergence to a local minimum of a degree-four polynomial is known to be NP-hard [MK87]. Several attempts have been made to solve the intensity-based optimization problem (4.7) where $\hat{Y}_{i,j}$ is replaced with $|F(X)|^2_{i,j}$, and the constraint $\hat{Y}_{i,j} = |F(X)|^2_{i,j}$ is removed. Such methods include the PhaseLift [CSV13] and Wirtinger Flow [CLS15] algorithms. In both instances, the authors establish provable results and convergence guarantees when the sample measurements are Gaussian, but not Fourier. Establishing provable results when the sampling measurements are Fourier-based is still, to our knowledge, an open problem.

If we instead assume that our measured images suffer from Poisson shot noise, then the probability of capturing the measured intensity $Y_{i,j}$ at pixel index (i,j), given the estimate $|F(\hat{X})|^2_{i,j}$, can be expressed as

$$p(Y_{i,j} \mid |F(X)|^2_{i,j}) = \frac{\hat{Y}_{i,j}^{Y_{i,j}} \exp(-\hat{Y}_{i,j})}{Y_{i,j}!} \approx \frac{1}{\sqrt{2\pi N_{i,j}}} \exp\left(-\frac{(Y_{i,j} - \hat{Y_{i,j}})^2}{2N_{i,j}}\right). \tag{4.8}$$

From the central limit theorem, when the expected value of the Poisson distribution is large, then

this Poisson distribution will become more like a Gaussian distribution. Moreover, the resulting standard deviation will be proportional to the square root of the intensity measurement, $N_{i,j} \propto \sqrt{Y_{i,j}}$. This means that a large measured intensity at a pixel will imply large noise at that pixel. Under this assumption, which is only true in the limit of infinite photos, we can take the negative log-likelihood of the maximum likelihood estimator, and this time arrive at the weighted least-squares optimization problem

$$\begin{aligned} \min_{X} \quad & \sum_i \sum_j \frac{(Y_{i,j} - \hat{Y}_{i,j})^2}{2N_{i,j}^2}. \\ \text{subject to} \quad & \hat{Y}_{i,j} = |F(X)|_{i,j}^2, \quad \forall i,j \\ & X \in \mathcal{D} \end{aligned} \tag{4.9}$$

Minimizing this objective, rather than the former, may help to combat any heterogeneity within the data. Waller et al. [YDZ+15] provide an excellent comparison of this, and other variants to solving the phase retrieval problem (4.4). However, their main application is to super-resolution imaging, which is introduced and discussed in the next chapter.

Methods like the PhaseLift algorithm [CSV13] reformulate the phase retrieval problem into a semidefinite program by *lifting* the problem into a higher-dimensional space, and solving the corresponding convex relaxation exactly like the SDP relaxations we discussed in chapter 3. The authors prove that under certain conditions, and in the presence of noise, that they are still able to reconstruct the original image. The main assumption of their proof is Gaussian sampling vectors. This approach follows in the same vein as bilinear optimization and quadratic programming, and many techniques have been implemented to solve (4.4) without strong guarantees for Fourier-based measurements, in general.

4.3 Alternating Projection Algorithms for Phase Retrieval

Somewhat surprisingly, there is a class of nonconvex, iterative algorithms that achieve remarkable results for reconstructing an image up to some global phase ambiguity. In his 1982

comparison of phase retrieval algorithms [Fie82], Fienup showed empirically that one can reconstruct an image from Fourier measurements by applying a series of alternating projections between the image domain and Fourier domain. In his analysis, defining $\mathcal{D}_0 = \{x | x \in \mathbb{R} \text{ and } x \geq 0\}$, $D = D_0 \otimes D_0 \otimes \ldots \otimes D_0$ is the image domain and $\mathcal{F} = \{x \mid |F(x)| = y\}$ is the Fourier domain, where y is the measured Fourier modulus, or square root of the measured intensity. Let X_k denote the estimated image at iteration k and $\hat{X}_k$ its Fourier transform. Fienup compared two algorithms, the Error-Reduction (ER), or Gerchberg-Saxton algorithm, and his novel Hybrid Input-Output (HIO) algorithm.

The Error-Reduction algorithm consists of four basic steps as follows. At iteration k:

(1) Take the Fourier transform of X_k, $F(X_k) = \hat{X}_k$

(2) Replace $|\hat{X}_k|$ with the true Fourier modulus data, Y

(3) Take the inverse Fourier transform to get the new image, X'_k

(4) $X_{k+1} = \begin{cases} X'_{k_{ij}} & \text{if} \quad X'_{k_{ij}} \in D \\ 0 & \text{else} \end{cases}$

Defining the Fourier estimation error as

$$E^2_{F_k} = \frac{1}{N} \sum_u (|\hat{X}_k| - Y)^2, \tag{4.10}$$

and the image-domain estimation error as

$$E^2_{D_k} = \sum_x (|X'_k| - Y)^2 = \sum_{x \notin D} (X'_k)^2, \tag{4.11}$$

one can show that both error measurements are reduced at every iteration using Parseval's theorem.

Fienup's nonconvex Hybrid Input-Output algorithm is similar to the ER algorithm, differing only in the final step where the image-domain constraint is enforced. Keeping the first three steps intact, step 4 is replaced by

$$X_{k+1} = \begin{cases} X'_{k_{ij}} & \text{if } X'_{k_{ij}} \in D \\ X_{k_{ij}} - \beta X'_{k_{ij}} & \text{else} \end{cases} \tag{4.12}$$

where β is a positive constant. The first three steps ensure that the estimated Fourier measurements are feasible; however, unlike the ER algorithm, the HIO algorithm sacrifices feasibility in the image-domain and instead remodels the problem in order to provide negative feedback to the system. That is, if $X'_{k_{ij}} < 0$ for more than one iteration, then the corresponding pixel continues to grow larger and larger until it eventually becomes non-negative. Since the input, X_k, into the next iteration does not satisfy the image-domain constraints, using $E^2_{F_k}$ as an error measure is meaningless (it isn't a *true* estimate any longer). Therefore, $E^2_{D_k}$ is used to determine convergence, or termination of the algorithm. While we aren't guaranteed a reduction of error at every iteration, HIO can be shown to converge much more quickly than the ER algorithm and works very well in practice.

Bauschke et al. [BCL03] recently analyzed the HIO algorithm and showed that it was a non-convex instance of the Douglas-Rachford projection algorithm. The convergence of this algorithm is well understood, provided that all constraint sets are convex. However, the convergence results do not carry over to the nonconvex setting of phase retrieval. These insights led to a novel projection-based algorithm, termed the Hybrid Projection-Reflection (HPR) algorithm. The formulation by Bauschke et al. also incorporates a relaxation parameter for added flexibility, and further modifies the last step wherein we calculate the new image estimate.

While solving the Phase Retrieval problem is intrinsically difficult, the true and estimated phase information is limited by the resolving power of the imaging device. Thus, the ability to accurately reconstruct an image from its modulus data is hindered by both the ill-posedness of the problem and the limitations of the imaging device. Without *a priori* information, we may not be able to get past the ill-posedness of the problem; however, there is a class of methods called super-resolution techniques that may allow us to obtain more information about the sampled object than by simply using the imaging device.

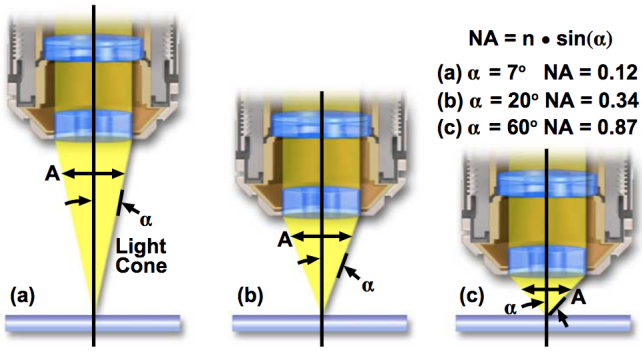
4.4 Super-Resolution Imaging

The resolution of an optical (or light) microscope is defined as the smallest distance between two points such that the points can still be distinguished as separate entities. There are many factors that can affect the resolving power of an optical system, including aberrations on the objective lens or systematic setup errors. However, there is a fundamental limit to the resolution of any imaging system, due to the physics of diffraction. The resolution of a microscope objective is determined by the angle of light waves that are able to enter the front lens, thus, the instrument is said to be diffraction-limited. This limit is purely theoretical, and it was first introduced by Ernst Abbe in 1873 [Abb73]. He found that light with wavelength λ, traveling in a medium with refractive index n, and converging to a spot with half-angle θ will have a minimum resolvable distance of

$$d = \frac{\lambda}{2n\sin(\theta)} = \frac{\lambda}{2\text{NA}}, \tag{4.13}$$

where $\text{NA} = n\sin(\theta)$ is the numerical aperture. This distance is known as the effective resolution of a microscope and in modern optics the numerical aperture can reach about 1.4-1.6.

Figure 4.1: Visualization of numerical aperture[1], with $n = 1$.



In order to minimize the resolvable distance between two points one can increase the size of the numerical aperture, decrease the wavelength of the light source, or both. Though these techniques offer better resolution, increasing the NA or using a light source with a shorter wavelength is

[1] Numerical Aperture and Resolution. *Olympus Scientific Solutions*, Retrieved from https://www.olympus-lifescience.com/en/microscope-resource/primer/anatomy/numaperture/

expensive and can be impractical for the application. As an example, consider violet light which has a wavelength $\lambda = 400$ nm and a NA of 1, the Abbe limit is roughly $d = \lambda/2 = 200$ nm (or 0.2 μm). This is small compared to most biological cells (1 μm to 100 μm), but large compared to viruses (100 nm), proteins (10 nm) and less complex molecules (1 nm). For biological samples, decreasing the wavelength of light to UV or X-ray may offer better resolution, but may damage the sample. Therefore, even a theoretically ideal objective without any imaging errors has a finite resolution.

4.4.1 Super-Resolution Techniques

There are techniques for reconstructing images that appear to have a higher resolution than allowed by the simple use of diffraction-limited optics. Super-resolution microscopy describes a class of techniques that can be categorized by two major groups: deterministic and stochastic.

Deterministic super-resolution involves ensemble imaging approaches that use patterned illumination to spatially modulate the fluorescence behavior of molecules. Fluorophores are a fluorescent chemical compound that can re-emit light upon excitation. They are sometimes used as a dye for staining certain structures. In biological microscopy, they show a nonlinear response to excitation and this response can be controlled such that not all of them emit simultaneously, thereby achieving subdiffraction limited resolution [HBZ10]. Methods in this group include the simulated emission depletion (STED) microscopy, ground state depletion (GSD) microscopy, reversible saturable optical linear fluorescence transitions (RESOLFT), structured illumination microscopy (SIM), and saturated structured illumination microscopy (SSIM) [HJC02, Gus05, HEJH05, KH99, HW94, Gue95, Gus00].

Stochastic super-resolution takes advantage of single-molecule imaging, using photoswitching or other mechanisms to stochastically activate individual molecules [HBZ10]. The chemical complexity of many molecular light sources can be used to make several close-by fluorophores emit light at separate times, making them resolvable in time. The measured positions of individual fluorophores are used to reconstruct the image with subdiffraction limited resolution. These meth-

ods include super-resolution optical fluctuation imaging (SOFI) and all single-molecule localization methods (SMLM) such as spectral precision distance microscopy (SPDM), SPDM with physically modifiable fluorophores (SPDMphymod), photo-activated localization microscopy (PALM or FPALM), stochastic optical reconstruction microscopy (STORM), and stimulated emission depletion (STED) microscopy, the winner of the Nobel Prize in Chemistry 2014 (along with single-molecule microscopy) [HGM06, BPS$^+$06, RBZ06, GER$^+$09, RBG$^+$08, HW94].

These methods are not restricted to biological applications. Even when the use of fluorophores is not viable, there are techniques to circumvent the optical space-bandwidth limitations of an imaging device to resolve images beyond theoretical diffraction limits. Such methods include structured illumination microscopy and Fourier ptychography [YDZ$^+$15, YBF$^+$18, HCO$^+$15, BMH$^+$17]. All super-resolution techniques share a common theme: multiple images of the same object are cleverly taken using a structured pattern or in a localized manner. The super-resolved image is then reconstructed using the known pattern information, or in the case of localization methods, the individual localized snapshots are stitched together in a coherent fashion, producing an image with finer details than simply using the imaging device alone.

As an example, consider the very simple construction of sampling an object multiple times without any band-limiting. We assume the output is only corrupted by additive white Gaussian noise with constant variance, σ^2,

$$Y_i = X + Z_i, \qquad Z_i \sim N(0, \sigma^2). \tag{4.14}$$

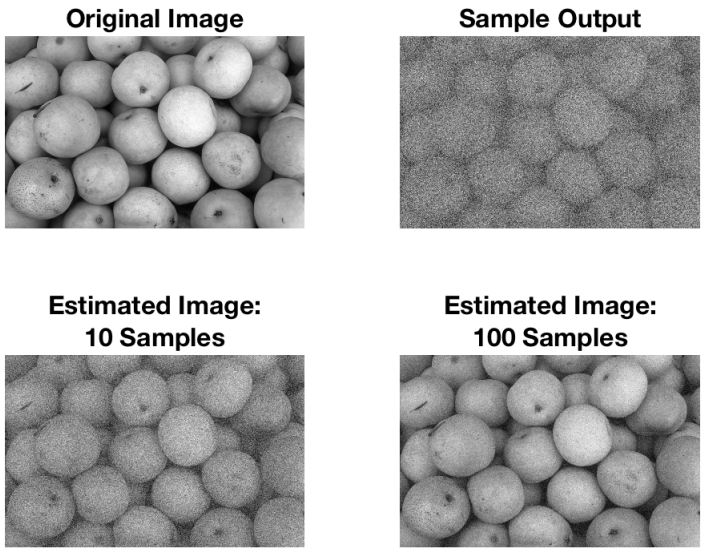
Since the noise has mean 0, an intuitive approach would be to sample the object multiple times and average over all the outputs to generate a suitable estimate for the true object. Assuming we have N samples,

$$\hat{X} = \frac{1}{N} \sum_{i=1}^{N} Y_i \approx X. \tag{4.15}$$

In Figure 4.2 we show an application of this example with a signal-to-noise ratio of 3.4. We see the estimate of the true image improves as we average over more samples.

What if our sampled images were instead corrupted with both additive and multiplicative

Figure 4.2: The top left subplot displays the original, uncorrupted image (`pears.png` from `MATLAB`'s Image toolbox). The top right subplot shows a sample output Y with a signal-to-noise ratio of 3.4. The bottom two images display the reconstruction after averaging over the specified number of samples taken.



noise? In that case, simply averaging over the outputs would not produce an adequate estimate. Let the underlying system be described by

$$Y = X \odot U + Z, \tag{4.16}$$

where $U, Z \sim N(0, \sigma^2)$ and $\odot$ represents element-wise multiplication. Figure 4.3 displays the results produced by averaging sample images that are contaminated with both additive and multiplicative noise. In this case, the effect of multiplicative white Gaussian noise renders the previous approach useless since

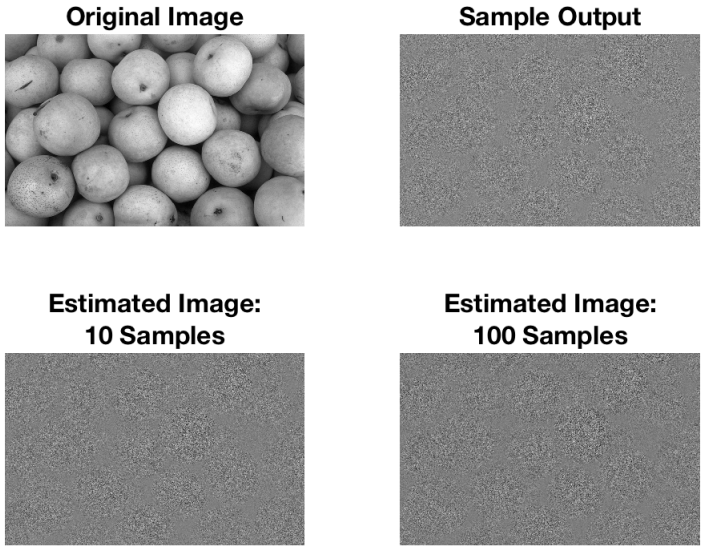
$$\hat{X} = \mathbb{E}[Y] = \mathbb{E}[X \odot U + Z] \approx X \odot \mathbb{E}[U] = 0. \tag{4.17}$$

However, since $U \sim N(0, \sigma^2)$, we can instead compute the following quantity

$$\hat{X}^2 = \mathbb{E}[Y^2] \approx \sigma^2 X^2, \tag{4.18}$$

where $(\cdot)^2$ represents element-wise squaring. Notice that by squaring each pixel value, the expec-

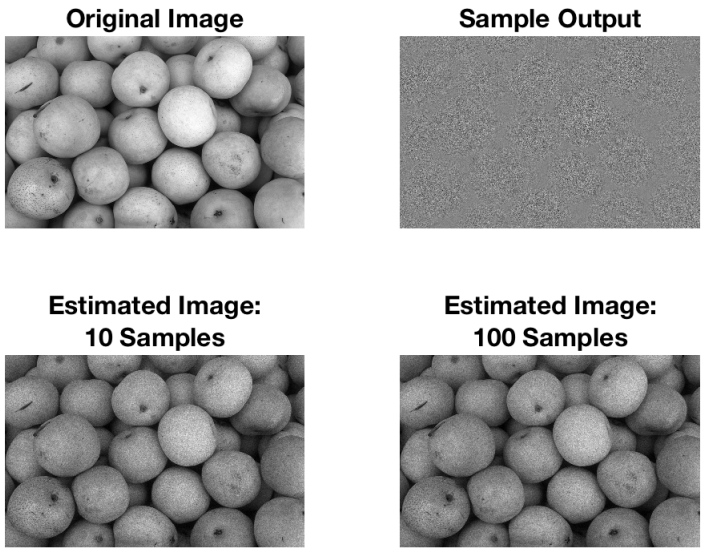
Figure 4.3: Comparing the results of averaging over sample images when we introduce both additive and multiplicative white Gaussian noise.



tation of U^2 is simply the variance. Therefore, we can estimate the image by averaging over the element-wise squared pixel values and dividing by the variance of the multiplicative noise. The reconstructed image is computed by taking the element-wise square root of the resulting estimate. The results of this procedure are displayed in Figure 4.4.

Clearly, (4.18) provides a better estimate for the true image than (4.17). Knowing the model describing the intrinsic system greatly improves the quality of the results since we may construct an inverse problem to compute an appropriate estimate for the true solution. Typically, the relationship between X and Y is more complex than that of the examples above. As such, we rely on the diversity amongst samples to identify patterns that might not otherwise be apparent. Ideally, our data would consist of a representative sample of the output space and our formulated model would be consistent with the underlying processes driving the system. Even in the additive noise case, the sample average is *not* the "best" thing to do: consider the James-Stein estimator. For this reason, and because in complicated systems we may not even *yet* have a clear idea of

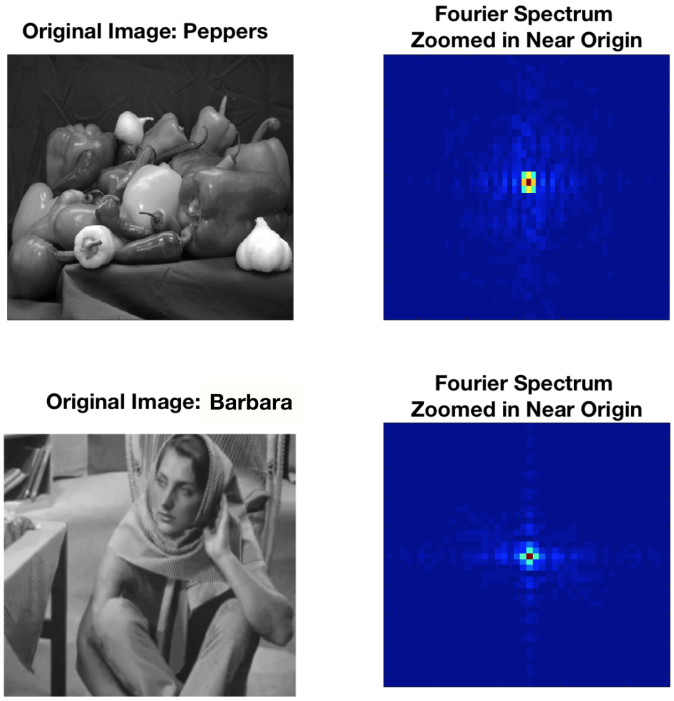
Figure 4.4: Comparing the results of averaging over element-wise squared pixel values and dividing by the multiplicative noise variance when we introduce both additive and multiplicative white Gaussian noise. The reconstructed image is computed by taking the element-wise square root of the resulting estimate. The results show the effect of considering 10 sample images versus 100.



the reconstruction, we're motivated to investigate the information gathering capabilities without having to actually form the estimate $\hat{X}$.

To understand the quantitative capabilities of super-resolution techniques, it is useful to think of sample structures as their Fourier representation. In this setting, low-frequency information is contained near the origin, and high-frequency information radially further away. A conventional microscope can only resolve images details with a line spacing larger than the diffraction limit of the objective lens, say d_0. Equivalently, it can only detect the information contained within a circular region of radius $1/d_0$ in Fourier space. This circular region defines the set of patterns (or observable frequencies) that the illumination light is able to generate. This observation is the basis for super-resolution techniques like SIM, and offers another means by which to understand the resolving power of an optical system.

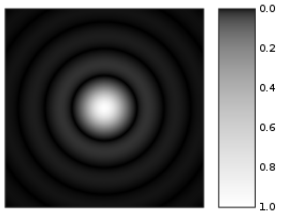
Figure 4.5: Examples of two images along with their Fourier transforms. The center of the Fourier spectrum represents the DC offset.



4.4.2 Regularization to Improve Image Reconstruction

In optical systems, the pixel pitch is the distance (in millimeters) from the center of a pixel to the center of the adjacent pixel. Thus, digital imaging devices are limited by the pixel pitch and pixel grid, as well as diffraction. The combined effect of the different components of an imaging system is described by the convolution of their point spread functions (PSF). Due to the physics of diffraction, and the way that light waves bend at a surface, the PSF of a diffraction limited lens is the Airy disk. These effects further limit the resolving power of an optical system and also limit the effectiveness of super-resolution techniques. As such, the methods mentioned above may not be sufficient as stand-alone techniques in reconstructing an image to a

Figure 4.6: Airy disk with grayscale intensities. It is the point spread function of a diffraction limited lens.



desired resolution.

To obtain a high-resolution reconstruction, one can append one or more penalty terms (or regularizers) to the negative log-likelihood to construct a regularized MLE. The goal of regularization is to ensure stability and accuracy in estimating the image. For instance, a recent paper [MHB$^+$17] investigated the use of spatially structured illumination in a wide-field fluorescence microscope to illuminate a sample with a series of excitation light patterns to generate a super-resolved image of two chromophores (the part of a molecule responsible for its color). The authors found that the ill-conditioning of their problem required them to impose a group-sparsity regularizer, as well as a quadratic penalty term known as the Tikhonov regularizer, to ensure resolution enhancement and stability of the solution.

It is worth noting that even if the structured illumination pattern is known, the resolution enhancement in structured illumination microscopy is limited to a factor of about two since the maximum spatial frequency of the illumination pattern is constrained by the microscope's optical transfer function, or Fourier transform of the the microscope's PSF [Gus00]. Therefore, there is a limit to the amount of information that can be gained from using these types of structured-illumination super-resolution techniques. Moreover, we pose the question: is the success of certain reconstruction algorithms due to the addition of prior information (i.e., regularizers), or due to the super-resolution imaging technique? We motivate this question with an example in the next section.

4.5 Recovering Image Statistics

Revisiting the regularized *blind structured illumination* problem from before [MHB$^+$17], Murray et al. were interested in reconstructing the distribution of the absorbed electromagnetic energy inside a biological sample. Their problem setup was as follows: given a measured signal $y(r)$ in photoacoustic microscopy, a PSF for the ultrasound detector (the transducer) $h(r)$, and local light fluence (radiation incident on a surface) $I(r)$, they model the measurements as

$$y(r) = h(r) \circledast [I(r) \cdot \rho(r)], \tag{4.19}$$

where $\circledast$ represents the convolution operator, $\cdot$ denotes point-wise multiplication, and $\rho(r)$ is the local absorption coefficient and the quantity of interest. In blind structured illumination, we use a series of **unknown** patterns satisfying some known statistical properties to down-shift some of the high-frequency components of the sampled image into the region of observable frequencies. This approach differs from the standard structured illumination technique wherein the sample is overlaid with a set of **known** patterns that aim to extract higher frequency information. Knowing the set of patterns allows us to solve a corresponding inverse problem for the estimated *super-resolved* image, $\hat{X}$.

In their blind structured illumination approach, light is focused on a small region, being careful to excite and illuminate one chromophore sample at a time. Their approach uses multiple unknown speckle illumination patterns that follow a statistical property that *should* allow them to resolve the image beyond the diffraction limit of the measuring device [MBG$^+$12, MJK$^+$13]. For M unknown speckled illuminations, the measured signals are defined as

$$y_m = h \circledast [I_m \cdot \rho] + \epsilon_m \quad \text{for} \quad m = 1, \ldots, M,$$

where ϵ_m is the noise, or error, in the data and I_m are the unknown speckled patterns. Letting $p_m := I_m \cdot \rho$, and using *a priori* knowledge of the problem structure, their objective is to minimize the following quantity:

$$F(p) = \frac{1}{2} \sum_{m=1}^{M} \sum_{i=1}^{N} |h \circledast p_m(x_i) - y_m(x_i)|^2 + \alpha_1 \|p\|_{2,1} + \frac{\alpha_2}{2} \|p\|_2^2 \to \min_p \tag{4.20}$$

where α_1 and α_2 are tuning parameters for the group-sparsity and Tikhonov penalty terms, respectively, and the x_i is the ith grid point of x representing equidistant points in the imaging domain.

While their approach is successful in estimating the original image, we are interested under-

standing the amount of information gained from applying blind structured illumination. Since we know in advance that the true image is sparse, it is appropriate to append a sparsity-promoting regularizer to the objective function being minimized. However, we are interested in disentangling the effect of regularization from the effect of implementing this super-resolution technique of overlaying a coordinate-wise multiplicative pattern on the sample to be estimated. Therefore, we consider solving the following problem, without the unknown speckled patterns

$$\min_{\rho} \frac{1}{2} \sum_{m=1}^{M} \sum_{i=1}^{N} |h \circledast \rho(x_i) - y_m(x_i)|^2 + \alpha \|\rho\|_1. \tag{4.21}$$

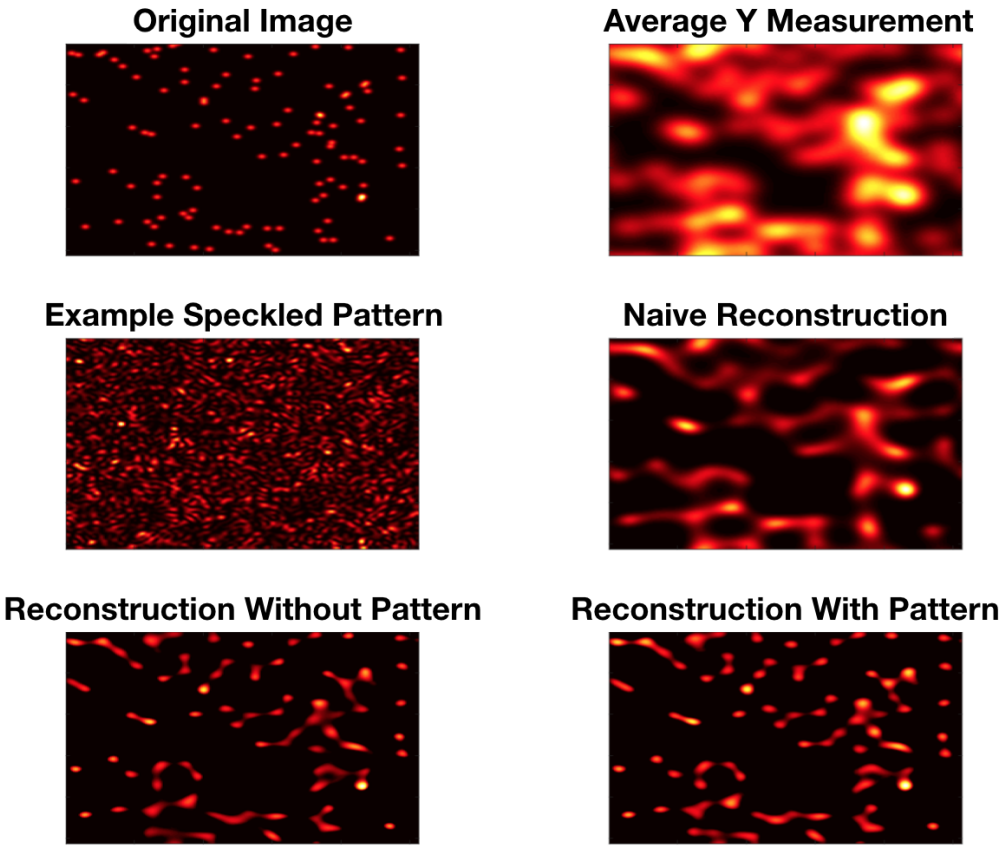
In this case, we are assuming each measurement, y_m, is expressed as

$$y_m = h \circledast \rho + \epsilon_m, \quad i = 1, \ldots, M.$$

In Figure 4.7 we compare the results of each technique [Mal]. For comparison, we include the result obtained by naively applying the Richardson-Lucy deconvolution method without any regularization, i.e., no addition of side information and speckled patterns. We see that with an appropriate choice of α, α_1, and α_2, we are able to reconstruct the original image up to some small error. By visual inspection, the two reconstructions appear very similar.

In this specific example, it seems as though regularization plays a significant role in estimating the original image, not the super-resolution technique. This is not to say that the technique offers us no advantage. Rather, we are interested in understanding the amount of information gained by taking multiple measurements under the blind structured illumination framework (and other super-resolution approaches) to quantify the benefit of implementing such a technique. We would like to circumvent the recovery process and apply information-theoretic concepts to recover statistics on the image generation process, before applying a reconstruction algorithm. In the next section we provide a brief overview of information-theoretic concepts to quantifying the uncertainty of a random variable, as well as the information contained between a pair of random variables. This will set up the foundation for our proposed method for quantifying the efficacy of various super-resolution techniques.

Figure 4.7: Comparison of various approaches to solving the photoacoustic microscopy problem. We display the original image to be estimated along with an example of a typical speckled pattern. The result of simply averaging over all measurements without a speckled pattern, but with additive noise is shown in the top right subplot. The middle right image shows the result of applying the Richardson-Lucy deconvolution algorithm on the set of images obtained without using a speckled pattern and without any type of regularization. The bottom two subplots display the results of implementing (4.20) and (4.21).



4.5.1 Information Theory Preliminaries: Mutual Information and Channel Capacity

At their most fundamental levels, the quantities of information theory – entropy, relative entropy, and mutual information – are defined as functionals of probability distributions [CT12]. Information theory studies the quantification, storage, and communication of information. We begin by defining a few useful concepts.

Definition 4.5.1. The *differential entropy* $h(X)$ of a continuous random variable X with density

$f(x)$ is defined as

$$h(X) = -\int_S f(x) \log f(x) dx, \tag{4.22}$$

where S is the support set of the random variable, *if it exists*.

The entropy of a random variable is a concave function that measures the amount of uncertainty of the random variable; it is the amount of information required on average to describe the RV. In the discrete case, $f(x)$ is replaced with the probability mass function, $p(x)$, and the integral is replaced by a summation. In both cases, the entropy depends only on the probability distribution of the random variable, and not the actual values of the RV.

Definition 4.5.2. If $X,\ Y$ have a joint density function $f(x, y)$, we can define the joint and conditional differential entropies as

$$\begin{aligned} h(X, Y) &= -\iint f(x, y) \log f(x, y)\ dxdy \\ h(X|Y) &= -\iint f(x, y) \log f(x|y)\ dxdy \end{aligned} \tag{4.23}$$

Since in general $f(x|y) = f(x, y)/f(y)$, we can also write

$$h(X|Y) = h(X, Y) - h(Y). \tag{4.24}$$

Thus, we can think of the entropy of a pair of RVs as the entropy of one plus the the conditional entropy of the other.

Definition 4.5.3. The *relative entropy* (or *Kullback-Leibler distance*) $D(f||g)$ between two densities f and g is defined by

$$D(f||g) = \int f \log \frac{f}{g}. \tag{4.25}$$

We note that $D(f||g)$ is finite if and only if the support set of f is contained in the support set of g. In statistics, the relative entropy arises as the expected logarithm of the likelihood ratio; it is a measure of the inefficiency of assuming that the distribution is q when the true distribution is p.

Definition 4.5.4. The *mutual information* $I(X;Y)$ between two random variables with joint density $f(x,y)$ is defined as

$$I(X;Y) = \int f(x,y) \log \frac{f(x,y)}{f(x)f(y)} dxdy \tag{4.26}$$

From the definition it is clear that

$$I(X;Y) = h(X) - h(X|Y) = h(Y) - h(Y|X) = h(X) + h(Y) - h(X,Y) \tag{4.27}$$

The mutual information is a measure of the amount of information that one random variable contains about another random variable; it is the reduction in uncertainty of one RV due to the knowledge of another. In terms of relative entropy, the mutual information $I(X;Y)$ is the relative entropy between the joint distribution and the product of marginal distributions, meaning, it measures the deviation from independence.

Theorem 4.5.1. *Let $X, Y \sim p(x,y) = p(x)p(y|x)$. The mutual information $I(X;Y)$ is a concave function of $p(x)$ for fixed $p(y|x)$ and a convex function of $p(y|x)$ for fixed $p(x)$.*

Definition 4.5.5. Random variables X, Y, Z are said to *form a Markov chain in that order* (denoted by $X \to Y \to Z$) if the conditional distribution of Z depends only on Y and is conditionally independent of X. Specifically, X, Y, and Z form a Markov chain $X \to Y \to Z$ if the joint probability mass function can be written as

$$p(x,y,z) = p(z|y)p(y|x)p(x). \tag{4.28}$$

Theorem 4.5.2. *(Data-Processing Inequality) If $X \to Y \to Z$, then $I(X;Y) \geq I(X;Z)$.*

Corollary 4.5.2.1. *In particular, if $Z = g(Y)$, we have $I(X;Y) \geq I(X;g(Y))$.*

This means that no processing on Y, deterministic or random, can increase the information that Y contains about X.

Definition 4.5.6. We define the "information" *channel capacity* of a discrete memoryless (conditionally independent of previous channel inputs or outputs) channel as

$$C = \max_{p(x)} I(X;Y), \tag{4.29}$$

where the maximum is taken over all possible input distributions, $p(x)$.

In practice, the channel capacity is the highest rate in bits per channel use at which information can be sent with arbitrarily low probability of error. In the discrete setting, it is the log of the number of distinguishable signals and it demonstrates that one can choose a "non-confusable" subset of input sequences such that with high probability there is only one highly likely input that could have caused the particular output. In general, there is no closed form for the solution of channel capacity; however, for many simple channels it is possible to calculate.

Using the above concepts, and in particular the Data-Processing Inequality, we aim to quantify the information gained from the raw measurements obtained by applying super-resolution techniques, prior to implementing a reconstruction algorithm. By the Data-Processing Inequality, post-processing of the raw measurements should not increase the amount of information that the measured images Y contain about the sample X, on average. We include the remark "on average" since these information-theoretic quantities are defined as functionals on probability distributions. In particular, we propose using the mutual information to quantify the amount of information a measurement Y contains about a sample X; or rather, the reduction in uncertainty of X given a measurement Y obtained using some super-resolution technique.

Estimating the entropy, mutual information, and channel capacity for a pair of random variables is, in general, a very challenging task. Under certain imaging and noise models, we may be able to compute quantities such as the channel capacity in closed form. However, quantities such as the mutual information require knowledge of the joint probability distribution $p(X, Y)$ and the marginal distribution of the output space $p(Y)$, which are typically unknown. In what follows, we describe two approaches to estimating the mutual information: a popular strategy that discretizes the joint probability space and computes the appropriate statistics on the discretized space, and our novel proposal of implementing Monte Carlo methods to estimate the underlying distributions. We discuss the pros and cons of each approach and conclude with ideas for future work.

4.5.2 Discretization Approaches to Estimating the Mutual Information

In most cases, the joint pdf between the input and the output, $p(X, Y)$, is unknown. This fact leads to the intrinsic difficulty in calculating the mutual information. Given N i.i.d. samples from $p(X, Y)$, $\{x_i, y_i\}_{i=1}^N$, it is unclear whether the set of points are sufficient to model the underlying relationship between X and Y, and thus some type of regularization is necessary. When both X and Y are continuous random variables, our parameter space is the space of smooth probability density functions. Given N realizations, the maximum likelihood estimator that best represents the provided data is the sum of dirac-delta functions centered at each realization, which does not exist in the parameter space. Therefore, we must be very careful in how we define and estimate the necessary distributions.

Several discretization approaches have been proposed to estimate the mutual information that rely on the relationship

$$I(X;Y) = h(X) + h(Y) - h(X,Y) \tag{4.30}$$

and the Data-Processing Inequality. We introduce several of these techniques below, and refer the reader to [Pan03] for a more comprehensive discussion. The general idea is as follows: pick a sequence of functions S_N and T_N that preserve the nature of the underlying problem and use these to estimate the mutual information $I(S_N, T_N)$ within some accuracy given N input-output pairs. S_N and T_N can be thought of as parameterizations of $p(X, Y)$ that allow the model to increase in complexity and provide tighter lower bounds for the true mutual information as more data becomes available.

Method of Sieves Grenander's Method of Sieves is an optimization approach to estimating non-parametric problems when the parameter space is too large [GH82, CG85, Gre81]. Methods such as maximum likelihood or least-squares can fail when applied to non-parametric problems, as alluded to with the sum of dirac-delta functions example above. The objective of the approach is to optimize over a subset of the parameter space, and choose increasingly dense subsets as more data becomes available. This subset is thought of as a "sieve," and the "mesh size" of the sieve is

decreased until the underlying density is arbitrarily close to the true distribution, in the limiting sense.

Since the Data-Processing Inequality is unconstrained by the transformation on the random variable (no transformation on Y can give us more information about X), we can choose S_N and T_N to discretize $\mathcal{X}$ and $\mathcal{Y}$ into a finite number of points $m_{S,N}$ and $m_{T,N}$. That is, $S_N : \mathcal{X} \to \{1, \ldots, m_{S,N}\}$ and $T_N : \mathcal{Y} \to \{1, \ldots, m_{T,N}\}$. Thus, we are able to reduce our original infinite-dimensional parameter space to the $m_{S,N}m_{T,N}$-simplex where the joint distribution of random variables S_N and T_N is discrete on $m_{S,N}m_{T,N}$ points.

Given N data points, the "empirical measure" of the ith bin is

$$p_{N,i} = \frac{1}{N} \sum_{j=1}^{N} \delta_i(T_N(y_j)), \quad i = 1, \ldots, m_{T,N} \tag{4.31}$$

where δ_i is the probability measure concentrated at i. We can think of $T_N(y_j)$ as the operation of "binning" y_j into the ith bin of the discretized space. Using this measure as the empirical probability density, we define three popular estimators used to estimate the entropy

$$h(p) = -\sum_i p \log p \tag{4.32}$$

where the following logs are natural, unless otherwise stated. The first estimator is the maximum-likelihood estimator (ML) [AK01, SKvSB98]

$$\hat{h}_{MLE}(p_N) = -\sum_{i=1}^{m} p_{N,i} \log p_{N,i}. \tag{4.33}$$

For a fixed discretization, the ML estimator is the best in a worst-case sense as $N \to \infty$. That is, it is asymptotically minimax in the ℓ_2-norm [Rao01]. However, it is important to note that this, and the subsequent estimators, are biased. The effects of the bias and their implications on the estimators are discussed below. The second estimator considered is a bias-corrected version of the maximum-likelihood estimator, called the Miller-Madow estimator [Mil55]

$$\hat{h}_{MM}(p_N) = \hat{h}_{MLE}(p_N) + \frac{\hat{m} - 1}{2N}, \tag{4.34}$$

where $\hat{m}$ is the number of bins with non-zero p_N-probability. Lastly, there is the jackknifed version of the MLE [ES81]

$$\hat{h}_{JK}(p_N) = N \cdot \hat{h}_{MLE}(p_N) - \frac{N-1}{N} \sum_{j=1}^{N} \hat{h}_{MLE-j}, \tag{4.35}$$

where $\hat{h}_{MLE-j}$ is the maximum-likelihood estimator on all but the jth sample. Each of these estimators excel in different problem regimes. Assuming the number of data points N is much larger than the number of bins m ($N \gg m$), if the variance dominates the mean-square error then all three estimators are approximately equivalent because they all share the same asymptotic variance rate. If the bias dominates, then the bias-corrected estimators $\hat{h}_{MM}$ and $\hat{h}_{JK}$ are more effective for estimating the entropy. However, when $N \sim m$, all three of these estimators begin to break down; thus, we require a substantial amount of data to accurately estimate the joint probability distribution, and in many instances the amount of data necessary to accurately estimate $p(X, Y)$ is unobtainable.

For a fixed measure p and discretization m, by the Central Limit Theorem, the empirical measures p_N become asymptotically normal as $N \to \infty$. Furthermore, since $\hat{h}_{MLE}$ is a smooth function of p_N, $\hat{h}_{MLE}$ is asymptotically normal, and both the bias and variance decrease approximately as $\frac{1}{N}$ in magnitude [Bas59]. It is important to note that asymptotic variances rates vary smoothly across the space of joint distributions $p(x, y)$ whereas the bias rate depends only on the number of nonzero elements of p [Pan03]. Much work has been done to estimate the bias in order to subtract it out of the entropy estimate directly [Mil55, Car69, TP95, Vic00]. In most cases, the bias is estimated using some series expansion that is either not absolutely, or not everywhere convergent. In [Pan03], Paninski proposes an alternate method for estimating the bias that leads to a better approximation of the entropy of the empirical distribution p_N which we discuss below. For completeness, a well-known fact about estimating the entropy is the following

$$E_p(\hat{h}_{MLE}) \leq h(p). \tag{4.36}$$

That is, the conditional expectation of the ML estimator of h over p is always less than or equal to the true entropy. Equality holds when $h(p) = 0$, or the underlying distribution is supported on

a single point. Therefore, the bias of the ML estimator is negative everywhere, except for the case $h(p) = 0$. The proof is an application of Jensen's inequality and can be found in [AK01].

Using the fact that any smooth function of the empirical measures p_N will act like an affine function as $N \to \infty$ with probability one [Ser80], we can expand the entropy function around the true density p to deduce the expected bias. Let

$$\begin{aligned}\hat{h}_{MLE}(p_N) &= h(p_N) \\ &= h(p) + dh(p; p_N - p) + r_N(h, p, p_N) \\ &= h(p) + \sum_{i=1}^{m} (p_i - p_{N,i}) \log p_i + r_N(h, p, p_N).\end{aligned}$$

where $r_N(h, p, p_N)$ is the remainder term and $dh(p; p_N - p)$ is the functional derivative of h in the direction $p_N - p$. It can be shown that $r_N(h, p, p_N) = -D_{KL}(p_N; p)$, the Kullback-Leibler divergence

$$D_{KL}(p_N; p) = -\sum_{i=1}^{m} p_{N,i} \log \left(\frac{p_i}{p_{N,i}} \right). \tag{4.37}$$

Since

$$E_p \left(\sum_{i=1}^{m} (p_i - p_{N,i}) \log p_i \right) = 0, \tag{4.38}$$

we have

$$E_p(\hat{h}_{MLE}) - h = -E_p(D_{KL}(p_N; p)). \tag{4.39}$$

Since $D_{KL}(p_N; p) \geq 0$, this implies the bias of the MLE is negative. Miller [Mil55] used a similar expansion to show that for a fixed p and m,

$$B(\hat{h}_{MLE}) = -\frac{m-1}{2N} + o(N^{-1}). \tag{4.40}$$

Thus, the bias of the ML estimator, as well as the Miller-Madow and jackknifed estimators, decrease at a rate $\frac{1}{N}$ as $N \to \infty$.

In his derivation of a new estimator $\hat{h}$ [Pan03], Paninski expands on work done by Devore and Lorentz [DL93] that relates the entropy function to a Bernstein polynomial approximation. If the polynomial is close to h, in a suitable sense, the bias will be small. This can be achieved with

an appropriate choice of polynomial coefficients. The author empirically shows that their novel estimator is comparable to the previously mentioned estimators when the amount of samples is much larger than the number of bins, $N \gg m$. When $N \sim m$, their estimator, which they call the "Best Upper Bound" estimator $\hat{h}_{BUB}$, outperforms $\hat{h}_{MLE}$, $\hat{h}_{MM}$, and $\hat{h}_{JK}$, but still suffers from bias and requires further investigation. Thus, their estimator is more applicable for a wider range of problems. Nevertheless, estimating the mutual information is still a challenging on-going research topic [GKOV17, KSG04, GVSG15].

The fact that these estimators are negatively biased means that they underestimate the amount of uncertainty of the random variable; this does not mean that they overestimate the mutual information between two variables. In order to estimate the true mutual information, we estimate the mutual information over a discretized version of the joint pdf, yielding a lower bound estimate for the true quantity. We can consider parameterizations with smaller bin sizes to compute a more accurate estimate for the mutual information; however, these estimators are negatively biased, and thus require access to an immense amount of data. In the next section we propose a novel approach to estimating the mutual information that entails no gridding and may reduce or avoid the biases discussed above.

4.5.3 Monte Carlo Methods to Measure Mutual Information

Monte Carlo methods are a numerical approach to solving problems that rely on randomness to solve a problem. They are very useful in simulating systems, such as fluids and coupled biological models. In essence, any problem involving a probabilistic interpretation can be approached using Monte Carlo methods. While these methods vary in practice, the general procedure is as follows:

(1) Define the input domain

(2) Generate inputs randomly from a probability distribution over the domain

(3) Perform a deterministic computation on the inputs

(4) Collect the results

That is, for a given distribution on the input variable, $p(x)$, the idea is to draw samples $x_i \sim p(x)$ and generate their corresponding outputs $y_i \sim p(y|x_i)$ to simulate a sequence of N input-output pairs $\{(x_i, y_i)\}_{i=1}^{N}$ from their joint distribution $p(x, y)$. For simplicity, we assume that the associated model noise has constant variance, σ^2. The PSF depends on the lens, and in particular it depends on the numerical aperture of the optical system. Hence, for a high numerical aperture we have a tight PSF and are, therefore, able to resolve an image with higher frequency information and to finer detail.

The additive white Gaussian noise model is one of the simpler models used to describe an imaged object. Let X denote the object projected onto the 2D object plane, i.e., $X \in \mathbb{R}^{m \times n}$, H the 2D impulse response of an ideal bandpass filter with cut-off frequency W (i.e., the PSF whose Fourier transform is 0 for all frequencies greater than W in magnitude), and $Z \sim \mathcal{N}(0, \sigma^2)$ the white Gaussian noise. The output of this channel Y can be formulated as

$$Y = (X + Z) \circledast H, \tag{4.41}$$

For channels of a single Gaussian band-limited signal, we have a closed form equation for the channel capacity

$$C = W \log \left(1 + \frac{P}{N_0 W}\right), \tag{4.42}$$

where P is the average energy or power constraint,

$$\frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} X_{i,j}^2 \leq P, \tag{4.43}$$

and $N_0/2$ watts/Hz is the noise spectral density. In order to arrive at this quantity, one must assume that X is sampled according to the Shannon-Nyquist sampling theorem. That is, the band-limited signal must be sampled at a rate $\frac{1}{2W}$ to sufficiently reconstruct the original image from its samples [Nyq24, Sha49].

In principle, one could compute (or at least approximate) the marginal distribution of the output space, $p(y)$, by "marginalizing out" the x variable,

$$p(y) = \int_x p(y|x)p(x)dx = \mathbb{E}_X[p(y|x)], \tag{4.44}$$

provided the conditional distribution $p(y|x)$ is either known in closed form or can be accurately estimated. Thus, by examining the conditional probability of Y given a particular value of X ($p(y|X = x_i)$), we can average the conditional probability over the distribution of all values of X to calculate the marginal probability of Y. We can then estimate the mutual information using the following approximation

$$I(X;Y) = \iint p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{p(x_i,y_i)}{p(x_i)p(y_i)}\right), \tag{4.45}$$

where $(x_i, y_i)_{i=1}^N \sim p(x,y)$. By sampling from the joint distribution and using these values ($\{(x_i,y_i)\}_{i=1}^N$) to estimate the integrand with respect to the probability measure, we can circumvent the issue of knowing the joint distribution $p(x,y)$. Furthermore, we can use the well-known relationship $p(x,y) = p(y|x)p(x)$ to rewrite the logarithm term as

$$\log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \log\left(\frac{p(y|x)}{p(y)}\right). \tag{4.46}$$

Hence, we can approximate the mutual information between our sample X and its raw measurements Y by computing

$$I(X;Y) \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{p(y|x)}{p(y)}\right). \tag{4.47}$$

These concepts can be used to measure the effectiveness of super-resolution techniques by comparing the information retained between the X and Y, as well as the information gained from taking multiple measurements of the same sample, $I(X;Y_1,\ldots,Y_k)$.

While these quantities can provide us a quantitative measure for establishing claims of super-resolution, there are several challenges to estimating the mutual information using Monte Carlo simulations. If the underlying distribution is continuous, then directly attempting to compute the mutual information may be infeasible due to numerical overflow and underflow resulting from the number of, and the size of, the sampled images. Since in many applications it is reasonable to

assume a discrete set of input images, we can replace the continuous probability distribution on X with a discrete one. Thus, for a finite set of images $X \in \{x_1,\ x_2,\ \ldots, x_M\}$, we define the probability of observing any one of these images as $p(X = x_i) = \frac{1}{M},\ i = 1, \ldots, M$, dramatically reducing the complexity of the problem.

Another difficulty in estimating the MI arises from estimating the marginal distribution of the output space

$$\begin{aligned} p(y) &= \int_{x \in \mathcal{X}} p(x)p(y|x)dx \\ &= \int_{x \in \mathcal{X}} p(y|x)dp(x). \end{aligned}$$

We may apply Monte Carlo simulations in the same fashion mentioned above; however, several key questions arise: (a) For a continuous random variable X, how many samples N_X are required to estimate the marginal distribution within some error, and are we required to re-sample the space each time we compute the value $p(y_i)$? and (b) How do we evaluate the conditional probability $p(y_i|x_i)$? Considering the additive white Gaussian noise model (4.41), the inherent difficulty in estimating the quantity $p(y|x)$ originates from the correlation that is introduced once the noise is band-limited. By *smoothing* the noise information (discarding all frequency information larger than the band-limit W in magnitude), we can think of each noise-pixel value being an average of the surrounding pixel values. Therefore, we cannot rely on the original assumption of uncorrelated noise measures to compute the conditional relationship between X and Y; therefore, the correlation structure must be estimated as well. By considering a discrete input distribution on X, we can remedy the issue of introducing biases from inadequately sampling the input space. Ideally, the N_X-generated samples would provide a representative sample of the input space; however, this is seldom the case. Depending on the amount and type of biases introduced by estimating the marginal distribution $p(y)$ using a fixed sampling of input space N_X (for the continuous case), we may be required to re-sample the input space each time we evaluate $p(y)$, which significantly increases the expected computational time needed to estimate the mutual information. We include the dependence of N_X on X since this set of input-samples should be different from the samples used to estimate the mutual information, $\{(x_i, y_i)\}_{i=1}^{N}$. Although there are several challenges to

estimating the mutual information using Monte Carlo methods, we believe this approaches to establishing claims of super-resolution can significantly impact the field of imaging and warrants further investigation.

4.6 Discussion

In this chapter we reviewed a very important problem in engineering and physics. We discussed several methods employed to tackle the phase retrieval problem, including a nonconvex approach that performs remarkably well in practice. In section 4.4 we introduced the topic of super-resolution imaging wherein many researchers have proposed various methods for resolving an image with a resolution exceeding the intrinsic limitations of the imaging device. However, we note that there is a limit to the amount of information gained by applying such techniques, and there is a need for better quantifying the effectiveness of these procedures. We propose applying information theoretic concepts to establish claims of super-resolution. We are interested in measuring the amount of information a raw measurement Y contains about a sample X, where Y is obtained using a super-resolution technique. Moreover, we propose using these concepts to calculate the total amount of information gained by taking multiple measurements of the same sample in a structured manner.

Our novel proposition of using Monte Carlo methods to estimating the mutual information preserved by a system (analogous to a communication channel) was discussed in section 4.5.3. We identified some of the inherent difficulties in directly computing the mutual information using a set of input-output data pairs, and offer a strategy for remedying issues associated with estimating these quantities for continuous distributions. By considering a discrete set of input images, we can significantly reduce the complexity of estimating the mutual information. We examined a discretization approach to estimating the entropy, and thus the MI, known as Grenander's Method of Sieves, and found that the empirical methods used to estimate the entropy function suffer from negative bias. Thus, they underestimate the true uncertainty in a measurement, and the magnitude of the bias decreases at a rate $\frac{1}{N}$. When the number of data samples is on the order of the number

of bins, these estimators perform poorly; therefore, in order to accurately estimate the desired quantities, we require access to a massive amount of data, at times unreasonable. Though a difficult problem, we believe that using the mutual information to quantify the efficacy of super-resolution techniques could over an enlightening means for establishing claims of super-resolution, and warrants further investigation.

Bibliography

[Abb73] Ernst Abbe, about a new lighting device on the microscope, Archive for Microscopic Anatomy **9** (1873), no. 1, 469–480.

[Adj14] Assalé Adjé, Policy iteration in finite templates domain, Numerical Software Verification (NSV 2014), 2014.

[Adj15a] ———, Overapproximating the reachable values set of piecewise affine systems coupling policy iterations with piecewise quadratic lyapunov functions, arXiv preprint arXiv:1506.02857 (2015).

[Adj15b] ———, Policy iteration in finite templates domain, Electronic Notes in Theoretical Computer Science **317** (2015), 3–18.

[AG15a] A. Adjé and P.-L. Garoche, Automatic synthesis of k-inductive piecewise quadratic invariants for switched affine control programs, Computer Languages, Systems & Structures (2015), available online.

[AG15b] Assalé Adjé and Pierre-Loïc Garoche, Automatic synthesis of piecewise linear quadratic invariants for programs, Verification, Model Checking, and Abstract Interpretation (VMCAI), 2015, pp. 99–116.

[AG15c] Assalé Adjé and Pierre-Loïc Garoche, Automatic synthesis of piecewise linear quadratic invariants for programs, International Workshop on Verification, Model Checking, and Abstract Interpretation, Springer, 2015, pp. 99–116.

[AGG12] Assalé Adjé, Stéphane Gaubert, and Eric Goubault, Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis, Logical Methods in Computer Science **8** (2012), no. 1.

[AGG14] Assalé Adjé, Stéphane Gaubert, and Eric Goubault, Computing the smallest fixed point of order-preserving nonexpansive mappings arising in positive stochastic games and static analysis of programs, Journal of Mathematical Analysis and Applications **410** (2014), no. 1, 227 – 240.

[AGM15] Assalé Adjé, Pierre-Loc Garoche, and Victor Magron, Property-based polynomial invariant generation using sums-of-squares optimization, arXiv preprint arXiv:1503.07025 (2015), Submitted.

[AGM16] ———, A sums-of-squares extension of policy iterations, arXiv preprint arXiv:1503.08090 (2016), Submitted.

[AGW15] Assalé Adjé, Pierre-Loc Garoche, and Alexis Werey, Quadratic zonotopes:an extension of zonotopes to quadratic arithmetics, arXiv preprint arXiv:1411.5847 (2015), Submitted.

[AHJS00] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard, A branch and cut algorithm for nonconvex quadratically constrained quadratic programming, Mathematical Programming **87** (2000), no. 1, 131–152.

[AK01] András Antos and Ioannis Kontoyiannis, Convergence properties of functional estimates for discrete distributions, Random Structures & Algorithms **19** (2001), no. 3-4, 163–193.

[AM14] Assalé Adjé and Victor Magron, Polynomial template generation using sum-of-squares programming, arXiv preprint arXiv:1409.3941 (2014), Technical Report.

[Ans09] Kurt M Anstreicher, Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming, Journal of Global Optimization **43** (2009), no. 2, 471–484.

[APS10] Gianluca Amato, Maurizio Parton, and Francesca Scozzari, Deriving numerical abstract domains via principal component analysis, pp. 134–150, Springer, 2010.

[Bas59] Georgij P Basharin, On a statistical estimate for the entropy of a sequence of independent random variables, Theory of Probability & Its Applications **4** (1959), no. 3, 333–336.

[BB88] Jonathan Barzilai and Jonathan M Borwein, Two-point step size gradient methods, IMA journal of numerical analysis **8** (1988), no. 1, 141–148.

[BBC+08] Pierre Bonami, Lorenz T Biegler, Andrew R Conn, Gérard Cornuéjols, Ignacio E Grossmann, Carl D Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, et al., An algorithmic framework for convex mixed integer nonlinear programs, Discrete Optimization **5** (2008), no. 2, 186–204.

[BCC+03] Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival, A static analyzer for large safety-critical software, Prog. Lang. Design & Implementation, ACM Press, 2003, pp. 196–207.

[BCC+05] ———, Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software (invited chapter)., In The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones, LNCS, vol. 2566, Springer, 2005, pp. 85–108.

[BCL03] Heinz H Bauschke, Patrick L Combettes, and D Russell Luke, Hybrid projection–reflection method for phase retrieval, JOSA A **20** (2003), no. 6, 1025–1034.

[Bel09] Pietro Belotti, Couenne: a users manual, Tech. report, Technical report, Lehigh University, 2009.

[BFGH17] S. Bogomolov, G. Frehse, M. Giacobbe, and T. Henzinger, Counterexample-guided refinement of template polyhedra, 2017, To Appear.

[BGL] Pierre Bonami, Oktay Günlük, and Jeff Linderoth, Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods, Mathematical Programming Computation, 1–50.

[BHZ06] R. Bagnara, P. M. Hill, and E. Zaffanella, The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems, Quaderno 457, Dipartimento di Matematica, Università di Parma, Italy, 2006.

[BLN95] R. H. Byrd, P. Lu, and J. Nocedal, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Stat. Comp. **16** (1995), no. 5, 1190–1208.

[BLNZ95] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu, A limited memory algorithm for bound constrained optimization, SIAM Journal on Scientific Computing **16** (1995), no. 5, 1190–1208.

[BMH+17] P Burgholzer, TW Murray, M Haltmeier, E Leiss-Holzinger, and T Berer, Photoacoustic super-resolution microscopy using blind structured speckle illumination, Photons Plus Ultrasound: Imaging and Sensing 2017, vol. 10064, International Society for Optics and Photonics, 2017, p. 100642H.

[BNO03] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, Convex analysis and optimization, Athena Scientific, 2003.

[BPS+06] Eric Betzig, George H Patterson, Rachid Sougrat, O Wolf Lindwasser, Scott Olenych, Juan S Bonifacino, Michael W Davidson, Jennifer Lippincott-Schwartz, and Harald F Hess, Imaging intracellular fluorescent proteins at nanometer resolution, Science **313** (2006), no. 5793, 1642–1645.

[BRZH02] R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill, Possibly not closed convex polyhedra and the Parma Polyhedra Library, Static Analysis Symposium, LNCS, vol. 2477, Springer, 2002, pp. 213–229.

[BV04] Stephen Boyd and Lieven Vandenberghe, Convex optimization, Cambridge university press, 2004.

[Car69] AG Carlton, On the bias of information estimates., Psychological Bulletin **71** (1969), no. 2, 108.

[CAS12] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan, Taylor model flowpipe construction for non-linear hybrid systems, Real Time Systems Symposium (RTSS), IEEE Press, 2012, pp. 183–192.

[Cas15] Pedro M Castro, Tightening piecewise mccormick relaxations for bilinear problems, Computers & Chemical Engineering **72** (2015), 300–311.

[CC76] Patrick Cousot and Radhia Cousot, Static determination of dynamic properties of programs, Proc. ISOP'76, Dunod, Paris, France, 1976, pp. 106–130.

[CC77] Patrick Cousot and Rhadia Cousot, Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints, ACM Principles of Programming Languages, 1977, pp. 238–252.

[CC92] Patrick Cousot and Radhia Cousot, Comparing the Galois connection and widening/narrowing approaches to Abstract interpretation, invited paper, PLILP '92, LNCS, vol. 631, Springer, 1992, pp. 269–295.

[CC07] Robert Clariso and Jordi Cortadella, The octahedron abstract domain, Science of Computer Programming **64** (2007), no. 1, 115 – 139.

[CE12] Xin Chen and Ábraham Erika, Choice of directions for the approximation of reachable sets for hybrid systems, EUROCAST'11 (Berlin, Heidelberg), Springer-Verlag, 2012, pp. 535–542.

[CG85] Yun-Shyong Chow and Ulf Grenander, A sieve method for the spectral density, The Annals of Statistics (1985), 998–1010.

[CGG+05] Alexandru Costan, Stephane Gaubert, Eric Goubault, Matthieu Martel, and Sylvie Putot, A policy iteration algorithm for computing fixed points in static analysis of programs, Computer Aided Verification (CAV), Lecture Notes in Computer Science, vol. 3576, Springer, 2005, pp. 462–475.

[CH78] Patrick Cousot and Nicholas Halbwachs, Automatic discovery of linear restraints among the variables of a program, POPL'78, January 1978, pp. 84–97.

[Chv83] Vaŝek Chvátal, Linear programming, Freeman, 1983.

[CLS15] Emmanuel J. Candes, Xiaodong Li, and Mahdi Soltanolkotabi, Phase retrieval via wirtinger flow: Theory and algorithms, IEEE Transactions on Information Theory **61** (2015), no. 4, 1985?2007.

[coi16] Couenne, a solver for nonconvex minlp problems, 2016.

[Cou05] Patrick Cousot, Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming, VMCAI, Lecture Notes in Computer Science, vol. 3385, Springer, 2005, pp. 1–24.

[CQ15] Frank E Curtis and Xiaocun Que, A quasi-newton algorithm for nonconvex, nonsmooth optimization with global convergence guarantees, Mathematical Programming Computation **7** (2015), no. 4, 399–428.

[CS05] Michael Colón and Sriram Sankaranarayanan, Towards space-efficient linear relations analysis, 2005, Draft available by request.

[CSS03] Michael Colón, Sriram Sankaranarayanan, and Henny Sipma, Linear invariant generation using non-linear constraint solving, CAV, LNCS, vol. 2725, Springer, July 2003, pp. 420–433.

[CSV13] Emmanuel J Candes, Thomas Strohmer, and Vladislav Voroninski, Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming, Communications on Pure and Applied Mathematics **66** (2013), no. 8, 1241–1274.

[CT12] Thomas M Cover and Joy A Thomas, Elements of information theory, John Wiley & Sons, 2012.

[CZ08] Gary Chartrand and Ping Zhang, Chromatic graph theory, Chapman and Hall/CRC, 2008.

[DL93] Ronald A DeVore and George G Lorentz, Constructive approximation, vol. 303, Springer Science & Business Media, 1993.

[DPG+14] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, Advances in neural information processing systems, 2014, pp. 2933–2941.

[DS07] David Delmas and Jean Souyris, Astrée: from research to industry, Proc. 14th International Static Analysis Symposium, SAS 2007, LNCS, vol. 4634, Springer, Berlin, 2007, pp. 437–451.

[DS13] Evrim Dalkiran and Hanif D Sherali, Theoretical filtering of rlt bound-factor constraints for solving polynomial programming problems to global optimality, Journal of Global Optimization **57** (2013), no. 4, 1147–1172.

[ES81] Bradley Efron and Charles Stein, The jackknife estimate of variance, The Annals of Statistics (1981), 586–596.

[Fie82] J. R. Fienup, Phase retrieval algorithms: a comparison, Appl. Opt. **21** (1982), no. 15, 2758–2769.

[FLD+11] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, SpaceEx: Scalable verification of hybrid systems, Proc. CAV'11, LNCS, vol. 6806, SV, 2011, pp. 379–395.

[Fle13] Roger Fletcher, Practical methods of optimization, John Wiley & Sons, 2013.

[GB94] L. El Ghaoui and V. Balakrishnan, Synthesis of fixed-structure controllers via numerical optimization, Proceedings of the 33rd Conference on Decision and Control(CDC), IEEE, 1994.

[GB08] Michael Grant and Stephen Boyd, Graph implementations for nonsmooth convex programs, Recent advances in learning and control (2008), 95–110.

[GBSBS18] Jessica Gronski, Mohamed-Amin Ben Sassi, Stephen Becker, and Sriram Sankaranarayanan, Template polyhedra and bilinear optimization, Formal Methods in System Design (2018).

[GBY08] Michael Grant, Stephen Boyd, and Yinyu Ye, Cvx: Matlab software for disciplined convex programming, 2008.

[GER+09] Manuel Gunkel, Fabian Erdel, Karsten Rippe, Paul Lemmer, Rainer Kaufmann, Christoph Hörmann, Roman Amberger, and Christoph Cremer, Dual color localization microscopy of cellular nanostructures, Biotechnology journal **4** (2009), no. 6, 927.

[GG10] Colas Le Guernic and Antoine Girard, Reachability analysis of linear systems using support functions, Nonlinear Analysis: Hybrid Systems **4** (2010), no. 2, 250 – 262.

[GGTZ07] Stephane Gaubert, Eric Goubault, Ankur Taly, and Sarah Zennou, Static analysis by policy iteration on relational domains, European Symposium on Programming, Lecture Notes in Computer Science, vol. 4421, Springer, 2007, pp. 237–252.

[GH82] Stuart Geman and Chii-Ruey Hwang, Nonparametric maximum likelihood estimation by the method of sieves, The Annals of Statistics (1982), 401–414.

[GKOV17] Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath, Estimating mutual information for discrete-continuous mixtures, Advances in Neural Information Processing Systems, 2017, pp. 5986–5997.

[GPBG08] Eric Goubault, Sylvie Putot, Philippe Baufreton, and Jean Gassino, Static analysis of the accuracy in control systems: Principles and experiments, FMICS, LNCS, vol. 4916, Springer, 2008, pp. 3–20.

[Gre81] Ulf Grenander, Abstract inference, Tech. report, 1981.

[GS07] Thomas Gawlitza and Helmut Seidl, Precise fixpoint computation through strategy iteration, European Symposium on Programming (ESOP), Lecture Notes in Computer Science, vol. 4421, Springer, 2007, pp. 300–315.

[GS11] Thomas Martin Gawlitza and Helmut Seidl, Solving systems of rational equations through strategy iteration, ACM Trans. Program. Lang. Syst. **33** (2011), no. 3, 11:1–11:48.

[GSV08] Sumit Gulwani, Saurabh Srivastava, and Ramarathnam Venkatesan, Program analysis as constraint solving, PLDI, ACM, 2008, pp. 281–292.

[Gue95] John M Guerra, Super-resolution through illumination by diffraction-born evanescent waves, Applied physics letters **66** (1995), no. 26, 3555–3557.

[Gus00] Mats GL Gustafsson, Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy, Journal of microscopy **198** (2000), no. 2, 82–87.

[Gus05] ———, Nonlinear structured-illumination microscopy: wide-field fluorescence imaging with theoretically unlimited resolution, Proceedings of the National Academy of Sciences **102** (2005), no. 37, 13081–13086.

[GVSG15] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan, Efficient estimation of mutual information for strongly dependent variables, Artificial Intelligence and Statistics, 2015, pp. 277–286.

[HBZ10] Bo Huang, Hazen Babcock, and Xiaowei Zhuang, Breaking the diffraction barrier: super-resolution imaging of cells, Cell **143** (2010), no. 7, 1047–1058.

[HCO+15] Roarke Horstmeyer, Richard Y Chen, Xiaoze Ou, Brendan Ames, Joel A Tropp, and Changhuei Yang, Solving ptychography with a convex relaxation, New journal of physics **17** (2015), no. 5, 053044.

[HEJH05] Michael Hofmann, Christian Eggeling, Stefan Jakobs, and Stefan W Hell, Breaking the diffraction barrier in fluorescence microscopy at low light intensities by using reversibly photoswitchable proteins, Proceedings of the National Academy of Sciences **102** (2005), no. 49, 17565–17569.

[HGM06] Samuel T Hess, Thanu PK Girirajan, and Michael D Mason, Ultra-high resolution imaging by fluorescence photoactivation localization microscopy, Biophysical journal **91** (2006), no. 11, 4258–4272.

[HJC02] Rainer Heintzmann, Thomas M Jovin, and Christoph Cremer, Saturated patterned excitation microscopya concept for optical resolution improvement, JOSA A **19** (2002), no. 8, 1599–1609.

[HM97] J.W. Helton and O. Merino, Coordinate optimization for bi-convex matrix inequalities, IEEE Conf. on Decision & Control(CDC), 1997, p. 36093613.

[HW94] Stefan W Hell and Jan Wichmann, Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy, Optics letters **19** (1994), no. 11, 780–782.

[JGN+17] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan, How to escape saddle points efficiently, ICML, 2017.

[KH95] Raymond Knopp and Pierre A Humblet, Information capacity and power control in single-cell multiuser communications, Communications, 1995. ICC'95 Seattle,'Gateway to Globalization', 1995 IEEE International Conference on, vol. 1, IEEE, 1995, pp. 331–335.

[KH99] Thomas A Klar and Stefan W Hell, Subdiffraction resolution in far-field fluorescence microscopy, Optics letters **24** (1999), no. 14, 954–956.

[KK01] Sunyonga Kim and Masakazu Kojima, Second order cone programming relaxation of nonconvex quadratic optimization problems, Optimization methods and software **15** (2001), no. 3-4, 201–224.

[KRK05] Snezana Krusevac, Predrag Rapajic, and Rodney A Kennedy, Channel capacity estimation for mimo systems with correlated noise, Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE, vol. 5, IEEE, 2005, pp. 5–pp.

[KS03] Michal Kočvara and Michael Stingl, PENNON: A code for convex nonlinear and semidefinite programming, Optimization methods and software **18** (2003), no. 3, 317–333.

[KSG04] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger, Estimating mutual information, Physical review E **69** (2004), no. 6, 066138.

[LF08] Francesco Logozzo and Manuel Fähndrich, Pentagons: A weakly relational abstract domain for the efficient validation of array accesses, Symposium on Applied Computing (New York, NY, USA), SAC '08, ACM, 2008, pp. 184–188.

[Lib08] Leo Liberti, Introduction to global optimization, Ecole Polytechnique (2008).

[Lin05] Jeff Linderoth, A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs, Mathematical Programming **103** (2005), no. 2, 251–282.

[LS10] Ricardo Lima and EWO Seminar, Ibm ilog cplex-what is inside of the box?, Proc. 2010 EWO Seminar, 2010.

[LSJR16] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht, Gradient descent only converges to minimizers, Conference on Learning Theory, 2016, pp. 1246–1257.

[Mal] Osman Malik, personal communication.

[Mata] Mathworks Inc., fmincon, Cf. `https://www.mathworks.com/help/optim/ug/fmincon` viewed April 2017.

[Matb] ———, PolySpace design verifier, Cf. `http://www.mathworks.com/products/polyspace/` viewed April 2017.

[MBG+12] Emeric Mudry, Kamal Belkebir, J Girard, Julien Savatier, E Le Moal, C Nicoletti, Marc Allain, and Anne Sentenac, Structured illumination microscopy using unknown speckle patterns, Nature Photonics **6** (2012), no. 5, 312–315.

[MG07] Jiri Matousek and Bernd Gärtner, Understanding and using linear programming, Springer Science & Business Media, 2007.

[MHB+17] Todd W Murray, Markus Haltmeier, Thomas Berer, Elisabeth Leiss-Holzinger, and Peter Burgholzer, Super-resolution photoacoustic microscopy using blind structured illumination, Optica **4** (2017), no. 1, 17–22.

[Mil55] George Miller, Note on the bias of information estimates, Information theory in psychology: Problems and methods (1955).

[Min01a] Antoine Miné, A new numerical abstract domain based on difference-bound matrices, PADO II, LNCS, vol. 2053, Springer, May 2001, pp. 155–172.

[Min01b] ———, The octagon abstract domain, AST 2001 in WCRE 2001, IEEE, IEEE CS Press, October 2001, pp. 310–319.

[MJK+13] Junhong Min, Jaeduck Jang, Dongmin Keum, Seung-Wook Ryu, Chulhee Choi, Ki-Hun Jeong, and Jong Chul Ye, Fluorescent microscopy beyond diffraction limits using speckle illumination and joint support recovery, Scientific reports **3** (2013), 2075.

[MK87] Katta G Murty and Santosh N Kabadi, Some np-complete problems in quadratic and nonlinear programming, Mathematical programming **39** (1987), no. 2, 117–129.

[Mur83] Katta G Murty, Linear programming, vol. 60, Wiley New York, 1983.

[NNH99] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin, Principles of program analysis, Springer, 1999.

[NW06] J. Nocedal and S. Wright, Numerical optimization, 2nd ed., Springer, 2006.

[Nyq24] Harry Nyquist, Certain factors affecting telegraph speed, Transactions of the American Institute of Electrical Engineers **43** (1924), 412–422.

[Opt17] Gurobi Optimization, Inc.,gurobi optimizer reference manual, 2017, URL: http://www.gurobi. com (2017).

[Pan03] Liam Paninski, Estimation of entropy and mutual information, Neural computation **15** (2003), no. 6, 1191–1253.

[PB17] Jaehyun Park and Stephen Boyd, General heuristics for nonconvex quadratically constrained quadratic programming, arXiv preprint arXiv:1703.07870 (2017).

[PGGSJ10] C Pozo, G Guillén-Gosálbez, A Sorribas, and L Jiménez, A spatial branch-and-bound framework for the global optimization of kinetic models of metabolic networks, Industrial & Engineering Chemistry Research **50** (2010), no. 9, 5225–5238.

[PT07] Imre Pólik and Tamás Terlaky, A survey of the s-lemma, SIAM review **49** (2007), no. 3, 371–418.

[QBM12] Andrea Qualizza, Pietro Belotti, and François Margot, Linear programming relaxations of quadratically constrained quadratic programs, Mixed Integer Nonlinear Programming (2012), 407–426.

[Rao01] BLS Prakasa Rao, Cramer-rao type integral inequalities for general loss functions, Test **10** (2001), no. 1, 105–120.

[RBG+08] Jürgen Reymann, David Baddeley, Manuel Gunkel, Paul Lemmer, Werner Stadter, Thibaud Jegou, Karsten Rippe, Christoph Cremer, and Udo Birk, High-precision structural analysis of subnuclear complexes in fixed and live cells via spatially modulated illumination (smi) microscopy, Chromosome Research **16** (2008), no. 3, 367–382.

[RBZ06] Michael J Rust, Mark Bates, and Xiaowei Zhuang, Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm), Nature methods **3** (2006), no. 10, 793.

[RD08] Øyvind Ryan and Mérouane Debbah, Channel capacity estimation using free-probability theory, IEEE Transactions on Signal Processing **56** (2008), no. 11, 5654–5667.

[RG05] Ted K Ralphs and Menal Güzelsoy, The symphony callable library for mixed integer programming, The next wave in computing, optimization, and decision technologies, Springer, 2005, pp. 61–76.

[RVS16] Pierre Roux, Yuen-Lam Voronin, and Sriram Sankaranarayanan, Validating numerical semidefinite programming solvers for polynomial invariants, Static Analysis Symposium (SAS), Lecture Notes in Computer Science, vol. 9837, Springer, 2016, pp. 424–446.

[RW09] R. Rockafellar and R. Wets, Variational analysis, Springer Berlin Heidelberg, 2009.

[Sah17] N. V. Sahinidis, BARON 17.8.9: Global Optimization of Mixed-Integer NLPs,, 2017.

[San05] Sriram Sankaranarayanan, Mathematical analysis of programs, Ph.D. thesis, Stanford University, 7 2005.

[SCCK05] Arak Sutivong, Mung Chiang, Thomas M Cover, and Young-Han Kim, Channel capacity and state estimation for state-dependent gaussian channels, IEEE Transactions on Information Theory **51** (2005), no. 4, 1486–1495.

[SDI08a] Sriram Sankaranarayanan, Thao Dang, and Franjo Ivančić, A policy iteration technique for time elapse over template polyhedra, HSCC, LNCS, vol. 4981, Springer, 2008, pp. 654–657.

[SDI08b] ———, Symbolic model checking of hybrid systems using template polyhedra, TACAS, LNCS, vol. 4963, Springer, 2008, pp. 188–202.

[SDL12] Hanif D Sherali, Evrim Dalkiran, and Leo Liberti, Reduced rlt representations for nonconvex polynomial programming problems, Journal of Global Optimization **52** (2012), no. 3, 447–469.

[Ser80] RJ Serfling, Approximation theorems of, Mathematical Sratistics (1980).

[Ser09] Robert J Serfling, Approximation theorems of mathematical statistics, vol. 162, John Wiley & Sons, 2009.

[SF02] Hanif D Sherali and Barbara MP Fraticelli, Enhancing rlt relaxations via a new class of semidefinite cuts, Journal of Global Optimization **22** (2002), no. 1, 233–261.

[SGS14] Mohamed Amin Ben Sassi, Antoine Girard, and Sriram Sankaranarayanan, Iterative computation of polyhedral invariants sets for polynomial dynamical systems, IEEE Conference on Decision and Control (CDC), IEEE Press, 2014, pp. 6348–6353.

[Sha49] Claude Elwood Shannon, Communication in the presence of noise, Proceedings of the IRE **37** (1949), no. 1, 10–21.

[SIG07] Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta, Program analysis using symbolic ranges, SAS, LNCS, vol. 4634, 2007, pp. 366–383.

[SKvSB98] Steven P Strong, Roland Koberle, Rob R de Ruyter van Steveninck, and William Bialek, Entropy and information in neural spike trains, Physical review letters **80** (1998), no. 1, 197.

[SP99] Edward MB Smith and Constantinos C Pantelides, A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps, Computers & Chemical Engineering **23** (1999), no. 4, 457–478.

[SSCÁ16] Mohamed Amin Ben Sassi, Sriram Sankaranarayanan, Xin Chen, and Erika Ábraham, Linear relaxations of polynomial positivity for polynomial lyapunov function synthesis, IMA Journal of Mathematical Control and Information **33** (2016), 723–756.

[SSM04a] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna, Non-linear loop invariant generation using Gröbner bases, ACM Principles of Programming Languages (POPL), ACM Press, 2004, pp. 318–330.

[SSM04b] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna, Constraint-based linear-relations analysis, Static Analysis Symposium (SAS 2004), LNCS, vol. 3148, Springer, August 2004, pp. 53–69.

[SSM04c] ———, Constructing invariants for hybrid systems, HSCC, LNCS, vol. 2993, Springer, 2004, pp. 539–555.

[SSM05] ———, Scalable analysis of linear systems using mathematical programming, Verification, Model-Checking and Abstract-Interpretation (VMCAI 2005), LNCS, vol. 3385, January 2005.

[SSM06] ———, Fixed point iteration for computing the time-elapse operator, HSCC, LNCS, Springer, 2006.

[ST97] Hanif D Sherali and Cihan H Tuncbilek, New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems, Operations Research Letters **21** (1997), no. 1, 1–9.

[Stu99] Jos F Sturm, Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones, Optimization methods and software **11** (1999), no. 1-4, 625–653.

[TP95] Alessandro Treves and Stefano Panzeri, The upward bias in measures of information derived from limited data samples, Neural Computation **7** (1995), no. 2, 399–407.

[TS05] M. Tawarmalani and N. V. Sahinidis, A polyhedral branch-and-cut approach to global optimization, Mathematical Programming **103** (2005), 225–249.

[VB04] Arnaud Venet and Guillaume P. Brat, Precise and efficient static array bound checking for large embedded C programs, PLDI, ACM, 2004, pp. 231–242.

[Vic00] Jonathan D Victor, Asymptotic bias in information estimates and the exponential (bell) polynomials, Neural computation **12** (2000), no. 12, 2797–2804.

[WB06] Andreas Wächter and Lorenz T Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical programming **106** (2006), no. 1, 25–57.

[Wei97] Volker Weispfenning, Quantifier elimination for real algebra—the quadratic case and beyond, Applied Algebra and Error-Correcting Codes (AAECC) 8, 1997, pp. 85–101.

[Whi92] Douglas J White, A linear programming approach to solving bilinear programmes, Mathematical Programming **56** (1992), no. 1, 45–50.

[WL05] Zhenghong Wang and Ruby B Lee, New constructive approach to covert channel modeling and channel capacity estimation, International Conference on Information Security, Springer, 2005, pp. 498–505.

[XVZ15] Wei Xia, Juan Vera, and Luis F Zuluaga, Globally solving non-convex quadratic programs via linear integer programming techniques, arXiv preprint arXiv:1511.02423 (2015).

[XWQ14] Yunhai Xiao, Soon-Yi Wu, and Liqun Qi, Nonmonotone barzilai–borwein gradient algorithm for ℓ_1-regularized nonsmooth minimization in compressive sensing, Journal of scientific computing **61** (2014), no. 1, 17–41.

[Yak71] Vladimir A Yakubovich, S-procedure in nolinear control theory, Vestnik Leninggradskogo Universiteta, Ser. Matematika (1971), 62–77.

[YBF+18] Jiun-Yann Yu, Stephen R Becker, James Folberth, Bruce F Wallin, Simeng Chen, VV Narumanchi, Jian Xing, and Carol J Cogswell, Achieving super-resolution fluorescence imaging with focused spot illumination and non-negative inversions (conference presentation), Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing XXV, vol. 10499, International Society for Optics and Photonics, 2018, p. 1049907.

[YDZ+15] Li-Hao Yeh, Jonathan Dong, Jingshan Zhong, Lei Tian, Michael Chen, Gongguo Tang, Mahdi Soltanolkotabi, and Laura Waller, Experimental robustness of fourier ptychography phase retrieval algorithms, Opt. Express **23** (2015), no. 26, 33214–33240.

[ZSL11] Xiao Jin Zheng, Xiao Ling Sun, and Duan Li, Convex relaxations for nonconvex quadratically constrained quadratic programming: matrix cone decomposition and polyhedral approximation, Mathematical programming **129** (2011), no. 2, 301–329.