Maximizing a Sum of Sigmoids

Madeleine Udell Stephen Boyd

May 5, 2014

Abstract

The problem of maximizing a sum of sigmoidal functions over a convex constraint set arises in many application areas. This objective captures the idea of decreasing marginal returns to investment, and has applications in mathematical marketing, network bandwidth allocation, revenue optimization, optimal bidding, and lottery design. We define the sigmoidal programming problem and show how it arises in each of these application areas. We show that the general problem is NP-hard, and propose an approximation algorithm (using a branch and bound method) to find a globally optimal approximate solution to the problem. We show that this algorithm finds approximate solutions very quickly on problems of interest. To illustrate the power of this approach, we compute the optimal positions which might have allowed the candidates in the 2008 United States presidential election to maximize their vote shares.

1 Introduction

1.1 Overview

The ability to efficiently solve large classes of convex optimization problems has enabled many of the greatest advances in operations research, machine learning, and control. By posing problems as convex programs, researchers and practitioners are able to take advantage of standard and scalable solvers which allow them to quickly find a global solution to their problems. When confronted with a non-convex problem, researchers may be tempted to give up hope of finding a global solution, and instead rely on heuristics and local optimization procedures. However, the quality of the solution obtained in this manner is generally unknown.

In this paper, we define a class of non-convex, NP-hard problems which we call *sigmoidal programs*, and describe an algorithm to find provably optimal global solutions. A sigmoidal program resembles a convex program, but allows a controlled deviation from convexity in the objective function. The framework that we present for sigmoidal programming is general enough to capture a wide class of objective functions, and any convex constraint set.

Our algorithm for sigmoidal programming relies on the well-known branch and bound method for non-convex optimization (Lawler and Wood 1966, Balas 1968). We compute upper and lower bounds for the sigmoidal programming problem by relaxing it to a tractable convex program. These upper and lower bounds are used as the basis of a recursion that eventually converges to the global solution to the problem. Sigmoidal programming derives its speedy performance in practice from the ease with which the convex relaxation may be computed. The time required to compute a solution using our proposed algorithm is a small multiple of the time required to solve a linear program, for problems that are "almost" convex programs or for problems with a small number of constraints.

Our main contributions in this paper are the identification of SP as a broad problem class with numerous applications; a method for constructing concave envelopes of sigmoidal functions, which makes the branch and bound approach computationally feasible; and a self-contained proof of convergence. We have also released an associated software package, `sigopt`, which implements the algorithm described here.

1.2 Outline

We first define the class of functions and problems that can be optimized using sigmoidal programming. We move on to applications in §3, and give a few examples of domains in which sigmoidal programming may be useful, and we discuss related work in §4. In §5 we prove the problem class is NP-hard, and give some results on approximability of SP. We describe our method for solving sigmoidal programming problems in §6, and we report numerical results in §7. In Appendix A, we prove that our method always converges to the optimal solution in (worst-case) exponential time.

2 Problem definition

In this paper, we consider the *sigmoidal programming problem*

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & x \in \mathcal{C}, \end{array} \tag{1}$$

where $f_i(x) : [l, u] \to \mathbf{R}$ is a sigmoidal function for each i, and the variable $x \in \mathbf{R}^n$ is constrained to lie in a nonempty bounded closed convex set $\mathcal{C}$.

A continuous function $f : [l, u] \to \mathbf{R}$ is defined to be *sigmoidal* if it is either convex, concave, or convex for $x \leq z \in [l, u]$ and concave for $x \geq z$ for some parameter $z \in \mathbf{R}$ (see Figure 1).

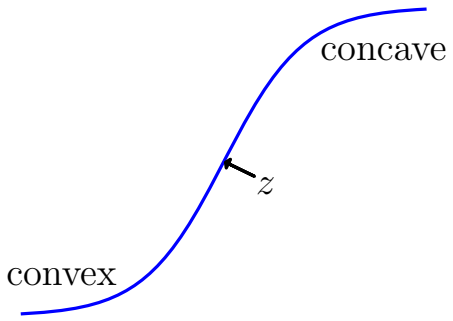

Figure 1: A sigmoidal function with inflection point at z.

2.1 Sigmoidal functions

Sigmoidal functions arise in a number of modeling contexts. We note first that all convex, concave, and affine functions are sigmoidal, according to our definition. The class also includes those functions whose graphs are "s-shaped", including the logistic function $\text{logistic}(x) = \exp(x)/(1 + \exp(x))$, and the error function $\text{erf}(x) = 2/\sqrt{\pi} \int_0^x \exp(-t^2)dt$. More generally, the cumulative distribution function (CDF) of any bounded quasi-concave probability distribution is sigmoidal. See Figure 2 for a few examples of sigmoidal functions.

Statistical models. Sigmoidal functions arise in the guise of CDFs in machine learning models for binary events. For example, if the probability of winning an auction or an election is fit using a logit or probit model, then that model gives the probability of winning as a sigmoidal function of the control parameters used to fit the model, such as money or time invested. The problem of winning a number of auctions or votes from separately modelled groups then becomes a sigmoidal programming problem.

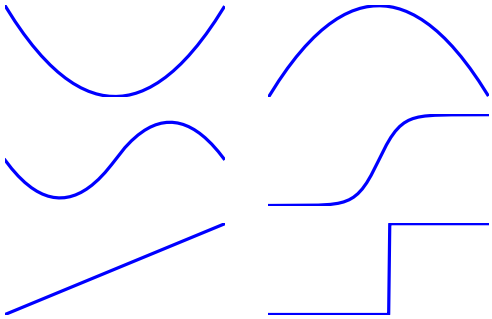

Figure 2: A few examples of sigmoidal functions.

Utility functions. In economics, the idea that curvature of the utility function might change sign dates back at least to Friedman and Savage (1948), who considered utility functions that were concave at low and high income levels, and convex in between (see Figure 3). They argued that the eponymous Friedman-Savage utility function might explain the willingness of a person to buy insurance for large losses (concave utility for losses) while also playing the lottery (convex utility for medium gains). The concavity of the utility function for extremely high gains was used to explain why lottery prizes are typically divided into many prizes of roughly similar size, as though to extend the income offered by the prize only to the upper limit of the convex portion of the utility curve. The Friedman-Savage utility function can be written as the sum of two sigmoids using the decomposition given below in equation (2). More recently, prospect theory has also posited sigmoidal utility functions for similar reasons (Kahneman and Tversky 1979, Tversky and Kahneman 1992). Hence the classical problem of societal utility maximization may be viewed as a sigmoidal programming problem.

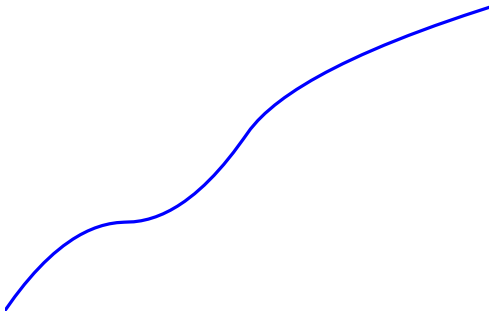

Figure 3: A Friedman-Savage utility function.

Step functions. Sigmoidal functions may also be used to approximate a step function to any arbitrary accuracy: for example, the *admittance* function

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x/\epsilon & 0 < x < \epsilon \\ 1 & x \geq \epsilon \end{cases},$$

which approximates a step function arbitrarily well as $\epsilon \to 0$, is sigmoidal for any $\epsilon > 0$. Sigmoidal functions that approximate a step function can be used to describe the utility from goods that are only useful in sufficient quantities, such as communications bandwidth for videoconferencing, or votes in a first-past-the-post election. Thus the problems of network bandwidth allocation and of winning elections in an electoral college system fit naturally in the sigmoidal programming paradigm.

Economies of scale. Sigmoidal functions aptly describe the profits of a firm that enjoys economies of scale as it increases to a moderate size and diseconomies of scale when it grows excessively large.

Everything else. It is worthwhile noting that while our definition of a sigmoidal function allows for only one inflection point, there is also a simple reduction of a known-inflection-point problem to a sigmoidal programming problem. Any function whose k inflection points are all known may be written as the sum of $k-1$ of sigmoidal functions.

For example, if $f : [l, u] \to \mathbf{R}$ is convex on $[l, z_1]$, concave on $[z_1, z_2]$, and convex on $[z_2, u]$, then f may be written as

$$f(x) = f_1(x) + f_2(x)$$

where f_1 and f_2 are both sigmoidal, *i.e.*,

$$\begin{aligned} f_1(x) &= \begin{cases} f(x) - 1/2 f'(z_2)(x - z_2) & x \leq z_2 \\ f(z_2) + 1/2 f'(z_2)(x - z_2) & x > z_2 \end{cases} \\ f_2(x) &= \begin{cases} 1/2 f'(z_2)(x - z_2) & x \leq z_2 \\ f(x) - f(z_2) - 1/2 f'(z_2)(x - z_2) & x > z_2 \end{cases} \end{aligned}$$

(see Figure 4). (Note that we must add a constraint requiring the arguments of f_1 and f_2 to be equal in order to fit the standard form SP.) It is also easy to search for inflection points z numerically using bisection.

Hence an algorithm for sigmoidal programming is effectively an algorithm for optimizing sums of functions with known curvature. Furthermore, sums of sigmoidal functions of the form

$$\sum_{i=1,\ldots,n} f_i(a_i y + b_i)$$

are dense in the space of continuous functions on a bounded domain $\mathcal{D} \subset \mathbf{R}^m$ (Cybenko 1989). Hence we can approximate any continuous function arbitrarily well by a suitably large linear combination of sigmoidal functions $f_i(x_i)$ if we add to the problem the affine constraint $x = Ay + b$ for some $y \in \mathbf{R}^m$.

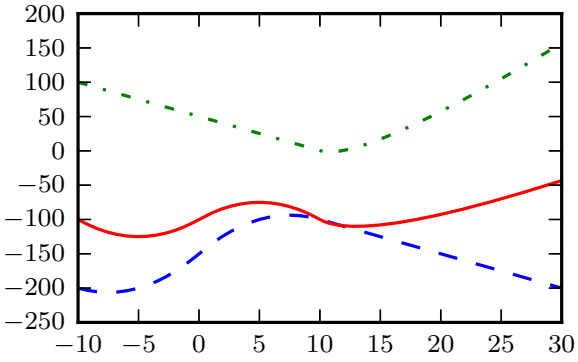

Figure 4: Decomposition of a function with two inflection points into two sigmoidal functions. The function f (solid line) is the sum of f_1 (dashed line) and f_2 (dot-dashed line).

3 Applications

In this section, we give a few applications of sigmoidal programming. The paradigm of sigmoidal programming is particularly well suited to solve *allocation problems*, in which a decision maker seeks to allocate a number of scarce resources between several objectives. Allocation problems arise naturally as a consequence of a decision maker's desire to maximize the positive outcomes of each of a number of different models, subject to resource constraints.

3.1 Machine learning models

For example, suppose an agent wishes to target each of n groups, with populations $p_i > 0$, $i = 1, \ldots, n$. The agent might be an advertising agency seeking to win market share, a political campaign seeking to win votes, a public health agency seeking to prevent disease, or a law enforcement agency seeking to detect criminal activity. The agent has access to a model for the efficacy of his actions on each group that depends on the quantity of a number m of scarce resources allocated to each group. This model gives the expected proportion $f_i(w_i^T y_i)$ of the group i for which the agent will be successful as a sigmoidal function f_i of a linear combination of the resources $y_i \in \mathbf{R}^m$ allocated to that group, where $w_i \in \mathbf{R}^m$ and $f_i : \mathbf{R} \to \mathbf{R}$ are parameters of the model, which we assume are given, and characterize the expected reaction of each market segment to the agent's actions. We may have a constraint on the total amount of each resource allocated,

$$\sum_{i=1}^{m} y_i \leq Y,$$

for some $Y \in \mathbf{R}^m$, and also a constraint on how much of each resource may be used for each group,

$$y^{\min} \leq y_i \leq y^{\max}, \quad i = 1, \ldots, n,$$

where $y^{\min}, y^{\max} \in \mathbf{R}^m$. The expected population that will be swayed by the agent's actions, summed over all groups, is

$$\sum_{i=1}^{n} p_i f_i(w_i^T y_i).$$

The problem is to choose y so as to maximize this quantity.

We can write this problem as a standard form sigmoidal programming problem using an auxiliary variable x whose ith component represents the linear combination of resources $w_i^T y_i$. The problem can be written as

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n p_i f_i(x_i) \\ \text{subject to} & \sum_{i=1}^m y_i \leq Y \\ & y^{\min} \leq y_i \leq y^{\max}, \quad i = 1, \ldots, n \\ & x_i = w_i^T y_i, \quad i = 1, \ldots, n. \end{array}$$

3.2 Cumulative distribution functions

Sigmoidal functions can arise as the cumulative distribution function of any quasi-concave probability distribution. We show in this section how to cast an optimal bidding problem as a sigmoidal programming problem.[1]

A bidder at an auction has a budget of B with which to bid on n different goods. The bidder privately believes that each good has a value v_i, $i = 1, \ldots, n$, and models the likelihood of winning the item as a sigmoidal function f_i of the bid b_i, $i = 1, \ldots, n$. The value derived by the bidder from a given bid is the expected profit from that bid, $(v_i - b_i) f_i(b_i)$. The bidder will not tolerate a negative expected profit, so we restrict our attention to bids $b_i \leq v_i$, $i = 1, \ldots, n$.

The problem is to maximize the total expected profit,

$$\sum_{i=1}^{n} (v_i - b_i) f_i(b_i),$$

subject to the limits on the bid values, $\sum_{i=1}^n b_i \leq B$ and $b_i \leq v_i$, $i = 1, \ldots, n$.

It is a simple exercise in univariate calculus to show that

$$(v_i - b_i) f_i(b_i)$$

is sigmoidal on the interval $(-\infty, v_i)$ if f_i is a sigmoidal CDF for every $i = 1, \ldots, n$.

Lottery design. We note that the opposite problem may also be of interest: that of designing a lottery or auction system that maximizes profit to the proprietor. Friedman and Savage (1948) argue that the curvature of the utility function implies that there is a unique

[1] We thank AJ Minich for developing this formulation of the optimal bidding problem in his class project for EE364b (Spring, 2011) at Stanford.

prize amount that lotteries ought to offer to maximize profit. The fact that lotteries often split their top prize into two or three equally sized prizes is evidence for their conjecture. Using sigmoidal programming, we can quantitatively test this theory. Suppose that all people have the same utility curve $f(x)$ as a function of income x. (This assumption makes no difference to the argument, but simplifies the notation.) The expected utility a person derives from a lottery ticket with prizes $x_i \geq 0$, $i = 1, \ldots, n$, and a cost per ticket $c > 0$ is

$$\mathbf{E}[f(x_i - c)] = \frac{1}{n} \sum_{i=1}^{n} f(x_i - c).$$

Participants will be willing to buy tickets for the lottery so long as the expected utility gain of participating is positive. The profit of the proprietor of the lottery is given by $nc - \sum_{i=1}^{n} x_i$. Hence a given profit P is feasible if the optimal value of the problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} f(x_i - c) \\ \text{subject to} & nc - \sum_{i=1}^{n} x_i \geq P \\ & x_i \geq 0, \quad i = 1, \ldots, n \\ & c > 0 \end{array}$$

is greater than zero. This problem can be transformed into a sigmoidal problem by decomposing the utility function into sigmoidal functions and introducing auxiliary variables, as was shown in §2.1.

The proprietor of the lottery maximizes his profit by finding the largest value of P such that the optimal value of (3.2) is positive, which can be computed by solving a sequence of sigmoidal programming problems to find a maximal P. Conversely, economists might use sigmoidal programming to infer the shape of lottery players' utility functions from a schedule of prizes.

3.3 Economies of scale

Sigmoidal functions also arise naturally in problems involving economies and diseconomies of scale. We give an example from revenue optimization.

A firm expects to enjoy economies of scale in the production of each of n goods, leading to increasing marginal returns as the amount of each good produced increases. However, the total market for each finished product is finite, and as the quantity of the good produced increases, the marginal return for each product will diminish or even become negative. The characteristic shape of the revenue curve is given by a function f_i for each good $i = 1, \ldots, n$, which is concave for large production volume, when the beneficial economies of scale are outweighed by the negative price pressure due to market saturation. The total expected revenue from product i if a quantity y_i of the good is produced is $f_i(y_i)$.

The quantity of each good produced is limited by the availability of each of m inputs, which might include money, labor, or raw materials. The plant has access to a quantity z_j of each input $j = 1, \ldots, m$, and can choose to allocate α_{ij} of each input j to the production of each good $i = 1, \ldots, n$, so long as $\sum_{i=1}^{m} \alpha_{ij} \leq z_j$. The firm requires γ_{ij} of input j for every

input $j = 1, \ldots, m$ to produce each unit of output i, so the total production y_i is always controlled by the limiting input,

$$y_i \leq \min_j \gamma_{ij}\alpha_{ij} \quad i = 1, \ldots, n.$$

The firm may also be subject to sector constraints limiting investment in each of a number of sectors either in absolute size or as a proportion of the size of the firm. These constraints can be written, respectively, as

$$Ay \leq b$$

or

$$Ay \leq \delta \mathbf{1}\mathbf{1}^T y,$$

where A is a matrix mapping outputs y into sectors, b gives absolute constraints on the size of each sector, δ is the maximum proportion of the total output that may be concentrated into any single sector, and $\mathbf{1}$ is the vector of all ones.

The total revenue of the firm can be written as

$$R = \sum_{i=1}^{n} f_i(y_i).$$

The problem is to choose y (subject to the input and sector constraints) so as to maximize R.

3.4 Network utility maximization

Sigmoidal programming has previously received attention as a framework for network utility maximization (NUM). Fazel and Chiang (2005) consider the problem of maximizing the utility of a set of flows through a network $G = (V, E)$, where the utility of a flow is a sigmoidal function of the volume of the flow, and each edge $e \in E$ is shared by many flows $i \in S(e)$ in the network. Here, we let x_i denote the volume of each flow $i = 1, \ldots, n$ and $f_i(x_i)$ the utility of that flow, while $c(e)$ gives the capacity of edge e. Fazel and Chuang consider only utilities that are expressible as polynomials f_i, which are required by their solution method. Using sigmoidal programming, one can solve NUM problems with polynomial utilities, but we can also solve problems in which utilities take other (sigmoidal) forms. We might take the utility to be an *admittance* function,

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x/\epsilon & 0 < x < \epsilon \\ 1 & x \geq \epsilon. \end{cases}$$

For example, the utility of network bandwidth to be used for videoconferencing or other realtime applications is nearly zero until a certain flow rate can be guaranteed, and saturates when the application is able to send data at the same rate as it is produced.

The NUM problem is to maximize the total utility of the flows $\sum_{i=1}^n f_i(x_i)$ subject to the bandwidth constraints $\sum_{i \in S(e)} x_i \leq c(e)$. Define the edge incidence matrix $A \in \mathbf{R}^{|E| \times n}$ mapping flows onto edges, with entries a_{ei}, $i = 1, \ldots, n$, $e \in E$. An entry $a_{ei} = 1$ if $i \in S(e)$, *i.e.*, if flow i uses edge e, and 0 otherwise. We write the NUM problem as

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & Ax \leq c \\ & x \geq 0. \end{array}$$

4 Related work

While our treatment of sigmoidal programming problems as a distinct problem class is new, our methods have deep historical roots. Below, we review some of the previous work for readers interested in a more thorough treatment of these topics.

Branch and bound. The branch and bound method for solving non-convex optimization problems has been well-known since the 1960s; see, for example, Balas (1968) for a brief overview of the branch and bound method applied to problems with discrete optimization variable, or Lawler and Wood (1966) for a more detailed review of the general method with continuous and discrete examples.

Nonconvex separable problems. A number of authors working before the advent of fast routines for convex optimization examined the possibility of using branch and bound techniques to solve nonconvex separable problems (McCormick 1976, Falk and Soland 1969). As in this paper, the authors suggested the use of convex subproblems based on convex envelopes as part of a larger branch and bound routine; but these papers did not give computationally efficient routines for computing the convex envelopes of the general univariate functions f_i they consider.

Convex optimization. In this paper, we treat convex optimization methods as an oracle for solving the subproblems that emerge in the iterations of the branch and bound method. That is, we compute the complexity of our algorithm in terms of the number of calls to a convex solver. The reader unfamiliar with convex optimization might consult Boyd and Vandenberghe (2004) for a comprehensive treatment of these methods and their applications. Linear programming solvers suffice as oracles for the applications we describe, all of which have only linear constraints; however, our convergence results and the algorithm presented encompass more general convex constraints, such as second order cone (SOCP) or semidefinite (SDP) constraints. Many fast solvers for LP, SOCP, and SDP are freely available in a variety of programming languages. See, for example, cvxopt (Andersen et al 2013), SeDuMi (Sturm 1999), SDPT3 (Toh et al 1999), and GLPK (Makhorin 2006).

Sigmoidal programming. To our knowledge, this paper presents the first unified approach to sigmoidal programming, and the first use of the branch and bound technique to globally optimize sigmoidal programming problems by solving a sequence of convex subproblems. However, other authors have previously considered special cases of the problem. For example, Fazel and Chiang (2005) explore an algorithm to compute global bounds on the optimal value of sigmoidal programming problems in the particular case of network utility maximization (NUM), though their method requires objective functions to be polynomially representable. They compute an upper bound on the optimal value using a sum of squares decomposition (Parrilo 2003), and a lower bound via a method of moments relaxation (Lasserre 2001, Henrion and Lasserre 2005), which are both found by solving an SDP. The quality of the bounds is controlled by the degree of polynomial allowed in the SDP; they find a small degree generally gives fine accuracy. However, even for small degrees, the complexity of these SDPs grow so quickly in the number of variables in the problem that all but the simplest problems are prohibitively expensive to solve. For example, the largest numerical example in (Fazel and Chiang 2005) has only 9 variables and 7 constraints.

Difference of convex programmming. Difference of convex (DC) programming (Horst et al 2000) can also be used to solve sigmoidal problems. The DC technique allows global optimization of a difference of (multivariate) convex functions. It is easy to see that sigmoidal functions can be written in this form (using a decomposition technique similar to equation (2)). However, known algorithms for DC programming require the solution of a sequence of (possibly) exponentially many NP-hard subproblems (Horst et al 2000), whereas the algorithm for sigmoidal programming presented here requires, even in the worst case, only the solution of exponentially many convex (and hence polynomial-time computable) subproblems.

5 Complexity

Hardness of sigmoidal programming. It is easy to see that sigmoidal programing is NP-hard. For example, one can reduce integer linear programming,

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & Ax = b \\ & x \in \{0,1\}^n, \end{array}$$

which is known to be NP-hard (Karp 1972), to sigmoidal programming:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n g(x_i) = x_i(x_i - 1) \\ \text{subject to} & Ax = b \\ & 0 \leq x_i \leq 1, \quad i = 1, \ldots, n, \end{array}$$

where $g(x)$ is a function chosen to enforce a penalty on non-integral solutions, *e.g.*, $g(x) = x(x-1)$. Then the solution to the sigmoidal program is 0 if and only if there is an integral solution to $Ax = b$.

Approximation guarantees. However, sigmoidal programming works extremely well for problems with a small number of linear constraints. Suppose that

$$\mathcal{C} = \{x \mid Ax \leq b,\ Gx = h\},$$

with $A \in \mathbf{R}^{m_1 \times n}$, so there are m_1 linear inequality constraints, and $G \in \mathbf{R}^{m_2 \times n}$, so there are m_2 linear equality constraints. Let $m = m_1 + m_2$ be the total number of constraints.

In a related paper (Udell and Boyd 2014), the authors show that it is possible to obtain an approximate solution for SP in polynomial time whose quality depends only on the number of constraints and the non-convexity of the functions f_i, and not on the dimension n of the problem. They define the *nonconvexity* $\rho(f)$ of a function $f : S \to \mathbf{R}$ to be

$$\rho(f) = \sup_x (f(x) - \hat{f}(x)),$$

They suggest approximating a solution to SP by solving a convex relaxation of the original problem. The approximation error of this solution is bounded by

$$\sum_{i=1}^{\min(m,n)} \rho_{[i]},$$

where we define $\rho_{[i]}$ to be the ith largest of the nonconvexities $\rho(f_1), \ldots, \rho(f_n)$. Hence for problems that are close to convex (*i.e.*, $\rho_{[1]}$ is small), or with very few constraints or variables (*i.e.*, m or n is small), we can expect that we can find a good approximate solution in a very small number of iterations. We take advantage of this property in the algorithm presented below.

The ease of solving sigmoidal programming problems with a small number of constraints makes this approach particularly well suited to solve allocation problems with a small number of resources. Indeed, we will see in §7 that all of the numerical examples we consider (including a problem with 10,000 variables) can be solved to very good accuracy in a few tens of iterations.

6 Method

The algorithm we present for sigmoidal programming uses a concave approximation to the problem in order to bound the function values and to locate the local maxima that are high enough to warrant consideration. In the worst case we may solve exponentially many convex optimization problems, but frequently we find bounds that are sufficiently tight after solving only a small number of concave subproblems. Furthermore, we may terminate the algorithm at any time to obtain upper and lower bounds on the possible optimal value, along with a feasible point that realizes the lower bound.

Branch and bound. The branch and bound method (Lawler and Wood 1966, Balas 1968, Balakrishnan et al 1991) is a recursive procedure for finding the global solution to an optimization problem restricted to a bounded rectangle Q_{init}. The method works by partitioning the rectangle Q_{init} into smaller rectangles $Q \in \mathcal{Q}$, and computing upper and lower bounds on the value of the optimization problem restricted to those small regions. Denote by $p^\star(Q)$ the optimal value of the problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & x \in \mathcal{C} \cap Q \end{array} \tag{2}$$

for any rectangle $Q \in \mathcal{Q}$. The global optimum must be contained in one of these rectangles, so if $U(Q)$ and $L(Q)$ denote the upper and lower bounds on $p^\star(Q)$,

$$L(Q) \leq p^\star(Q) \leq U(Q),$$

then

$$\max_{Q \in \mathcal{Q}} L(Q) \leq p^\star(Q_{\text{init}}) \leq \max_{Q \in \mathcal{Q}} U(Q).$$

The method relies on the fact that it is easier to compute tight bounds on the function value over small regions than over large ones. Hence by *branching* (dividing promising regions into smaller subregions), the algorithm obtains global bounds on the value of the solution that become arbitrarily tight.

Bound. In sigmoidal programming, we quickly compute a lower and upper bound on $p^\star(Q)$ by solving a convex optimization problem. Let $Q = I_1 \times \cdots \times I_n$ be the Cartesian product of the intervals $I_1, \ldots, I_n$. For each function f_i, $i = 1, \ldots, n$ in the sum, suppose that we have a concave upper bound $\hat{f}_i$ on the value of f_i over the interval I_i,

$$\hat{f}_i(x) \geq f_i(x), \quad x \in I_i.$$

Let $x^\star(Q)$ be the solution to the problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n \hat{f}_i(x_i) \\ \text{subject to} & x \in \mathcal{C} \cap Q. \end{array}$$

Then

$$p^\star(Q) \leq \sum_{i=1}^n \hat{f}_i(x_i^\star(Q)) = U(Q),$$

which gives an upper bound $U(Q)$ on the optimal value. (Note that Problem 6 could be infeasible, in which case we know that the solution to the original problem cannot lie in Q.)

To construct a lower bound, note that the value of the objective at any feasible point gives a lower bound on $p^\star(Q)$, and so in particular,

$$L(Q) = \sum_{i=1}^n f_i(x_i^\star(Q)) \leq p^\star(Q).$$

Theorem 2 from Udell and Boyd (2014) guarantees that these bounds satisfy

$$U(Q) - L(Q) \leq \sum_{i=1}^{\min(m,n)} \rho_{[i]},$$

where

$$\rho(f) = \sup_{x \in Q}(f(x) - \hat{f}(x))$$

and $\rho_{[i]}$ to be the ith largest of the nonconvexities $\rho(f_1), \ldots, \rho(f_n)$. As the algorithm proceeds, the size of the rectangles decreases, and so $U(Q) - L(Q)$ also decreases.

Concave envelope. If we choose $\hat{f}_i$ to be concave, then subproblem (6) is a convex optimization problem, and can be solved efficiently (Boyd and Vandenberghe 2004). The tightest concave approximation to f is obtained by choosing $\hat{f}$ to be the *concave envelope* of the function f, which is defined as the pointwise infimum over all concave functions g that are greater than or equal to f. For a sigmoidal function, the concave envelope is particularly easy to calculate. We can write $\hat{f}$ of f piecewise as

$$\hat{f}(x) = \begin{cases} f(l) + \frac{f(w)-f(l)}{w-l}(x-l) & l \leq x \leq w \\ f(x) & w \leq x \leq u, \end{cases}$$

for some $w > z$ (see Figure 5). The point w can easily be found by bisection: if $x < w$, then the line from $(l, f(l)$ to $(x, f(x)$ crosses the graph of f at x (from lying above the graph to lying below); if $x > w$, it crosses in the opposite direction.

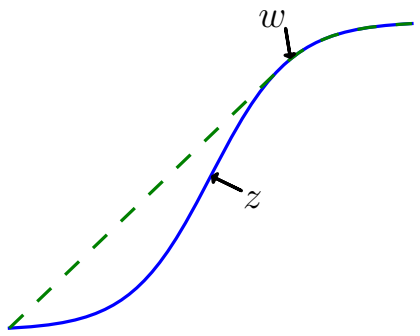

Figure 5: The concave envelope $\hat{f}$ (dashed line) of the logistic function f (solid line).

When all the constraints are linear, it may be convenient to construct a piecewise linear concave upper bound on the function f, and to maximize this piecewise linear upper bound instead of the concave envelope (see Figure 6). In this case, if $\mathcal{C}$ is a polyhedron, the

computation of upper and lower bounds on $p^\star(Q)$ reduces to a linear programming problem (Boyd and Vandenberghe 2004).

A piecewise linear bound $\bar{f}$ with $\sup_x \hat{f}(x) - \bar{f}(x) \leq \epsilon$ may be constructed as the minimum of the set of lines tangent to the graph of f at each point x_i in $S' = \{l\} \cup S$ for any $S \subseteq [w, \infty)$. (If $\hat{f}$ is not differentiable at $x_i \in S$, then any line tangent at x_i will work.) Since the maximum error $\hat{f}(x) - \bar{f}(x)$ in the approximation must occur where adjacent lines intersect, the points of intersection may be added to the set S' until the maximum error is less than ϵ.

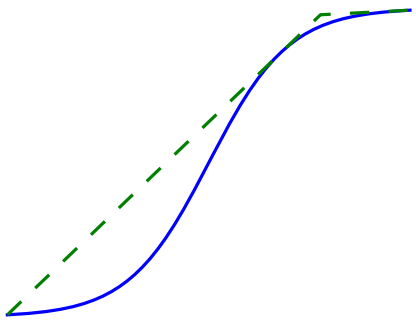

Figure 6: A piecewise linear approximation (dashed line) to the concave envelope of the (solid line) logistic function.

Branch. The concave envelope of f over the interval (l, u) is equal to f at l and u, and in general will lie closer to f on small intervals, or on intervals over which f is less strongly convex (closer to linear). These properties allow us to find tighter bounds on the optimal value of the problem over a smaller region than over a larger one, with the size required to find a bound of a given tightness controlled by the convexity of the function. Hence, we branch by splitting the rectangle with the largest upper bound along the coordinate i with maximum error $\epsilon_i = \hat{f}_i(x_i^\star(Q)) - f_i(x_i^\star(Q))$ into two subrectangles that meet at $x_i^\star(Q)$. The concave approximation of f_i on the subrectangles is then exact at $x_i^\star(Q)$, so that each branch maximally reduces the error at the previous solution.

Convergence. Each cut reduces the error by at least a factor of $1/n$, so only a finite number of cuts need be made before a given error tolerance is achieved. In Appendix A, we use this ideas to show that the number of concave subproblems that must be solved to achieve accuracy ϵ^{des} is bounded by

$$\prod_{i=1}^{n} \left(\left\lfloor \frac{(h_i(z_i) - h_i(l_i))\,(z_i - l_i)}{\epsilon^{\text{des}}/n} \right\rfloor + 1 \right),$$

where $Q_{\text{init}} = (l_1, u_1) \times \cdots \times (l_n, u_n)$, $z_i = \arg\max_{[l_i,u_i]} h_i(x)$ for $i = 1, \ldots, n$, and $f_i(x) = \int_{l_i}^{x} h_i(t)\, dt$ for some upper semi-continuous quasi-concave function $h_i : \mathbf{R} \to \mathbf{R}$ (which always exists, if f_i is sigmoidal).

Pruning. If $\max_{Q \in \mathcal{Q}} L(Q) > U(Q')$, we know with certainty that the solution is not in rectangle Q', and we may safely delete rectangle Q' from the list. In this case we say that rectangle Q' has been *pruned*. Conversely, we use the notion that a rectangle is *active* to mean that the possibility of finding the optimum in that rectangle has not been ruled out. The list of active rectangles maintained by the branch and bound algorithm then has an interpretation as the set of rectangles in which one is guaranteed to find at least one point for which the value of the objective differs from the optimal value by no more than $\max_{Q \in \mathcal{Q}} U(Q) - \max_{Q \in \mathcal{Q}} L(Q)$.

6.1 Extensions

Sums of sigmoidal functions. It is worthwhile noting that the algorithm given above may be applied to other problem classes. The only requirement of the algorithm is the ability to construct a concave upper bound on the function to be maximized on any rectangle, which becomes provably tight as the size of the rectangle decreases. Hence this algorithm may be applied not only to sums of sigmoidal functions, but to sums of other functions with other convexity properties.

We showed in the introduction that any univariate function with known regions of convexity and concavity can be written as a sum of sigmoidal functions. However, applying this algorithm directly to the original function might result in more tight concave approximations, and hence a faster rate of convergence. Conversely, we might decompose a sigmoidal function into one convex and one concave function, and apply the algorithm directly to these. However, the concave envelope of the convex function over the region will never be tighter than the concave envelope of the sigmoidal function. Hence the algorithm is unlikely to converge more quickly.

Penalty functions. Our method also extends to problems of the form

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} f_i(x_i) + \phi(x) \\ \text{subject to} & x \in \mathcal{C}, \end{array}$$

where ϕ is a concave reward function.

7 Numerical examples

All experiments were conducted on a desktop computer with a 3 GHz Intel Core 2 Duo processor, running Mac OS X 10.8.

7.1 Bidding example

As our first numerical example we consider an instance of the bidding problem given in §3.2.

In Figure 7, results are shown for an example in which

$$f_i(b_i) = \text{logistic}(\alpha_i b_i + \beta_i) - \text{logistic}(\beta_i),$$

with v_i drawn uniformly at random from the interval $[0, 4]$, $\alpha_i = 10$, $\beta_i = -3v_i$, and $B = .2\sum_{i=1}^{n} v_i$, for $i = 1, \ldots, n$. Each graph represents one variable and its associated objective function (solid line) and concave approximation (dashed line) on the rectangle containing the solution. The solution $x_i^\star$ for each variable is given by the x coordinate of the red X. The rectangle containing the solution is bounded by the solid grey lines and the endpoints of the interval. For $n = 36$, the solution even after the first iteration is quite close to optimality, and the sigmoidal programming algorithm reaches a solution within $\epsilon = .01$ of optimality after solving only 17 convex subproblems. Notice that the solution lies along the concave part of the curve, or at the lower bound, for nearly every coordinate. This phenomenon illustrates a generic feature of allocation problems: the convex portion of the curve offers the best "bang-per-buck" (*i.e.*, utility as a function of the resources used), and so the optimal solution generally exploits this region fully or gives up on the coordinate entirely.

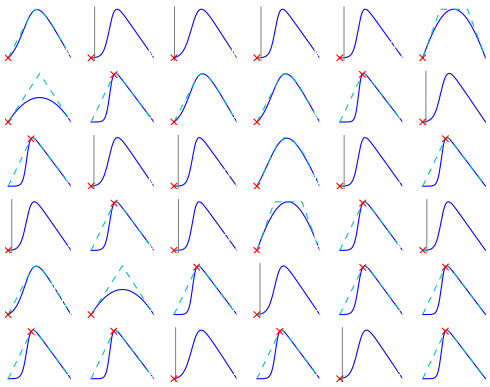

Figure 7: Solution for bidding example.

Table 1 gives the performance of the algorithm on bidding problems ranging in size n in producing a solution within $\epsilon = .01n$ of the global solution, with the other problem parameters drawn according to the same model as above. The table shows the average number of subproblems solved and time to achieve a solution averaged over 5 random problem instances. As n increases, the time to solve each problem increases (since the linear program that we solve at each step is substantially larger), but the number of subproblems solved stabilizes at two. One subproblem solve establishes a tentative solution, while the second verifies that the solution on the first subproblem is globally optimal by making the convex approximation tight at the previous solution.

Table 1: SP performance on bidding problems.

n	subproblems	time (s)
10	7.6	0.10
20	9.0	0.28
50	6.4	1.18
100	2.0	2.39
200	2.0	26.89
300	2.0	92.08
400	2.0	211.46
500	2.0	406.85

7.2 Network utility maximization

Our second example demonstrates the speed of the method on a typical allocation type problem. We consider the network utility maximization problem previously introduced in §3.4,

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax \leq c \\ & x \geq 0, \end{array}$$

where x represent flows, c are edge capacities, A is the edge incidence matrix mapping flows onto edges, and $f_i(x_i)$ is the utility derived from flow i.

Figure 8 shows the convergence of the algorithm for a network with $n = 500$ flows over 500 edges, with each flow using on average 2.5 edges each, where we use admittance functions as the utility functions f_i, and each edge has capacity 2.5. Within 14 iterations, the solution is within 3% of optimality.

7.3 Political marketing example

We now consider an application of the targeted marketing problem (§3.1) to political marketing, in which the decision variables correspond to the positions that a politician takes on each particular issue, and the functions correspond to the expected number of votes for that politician from a given constituency. The goal of the politician is to choose positions on each issue to maximize his vote share.

The idea that politicians might opportunistically choose their positions in order to maximize their vote share is very old, going back to Downs (1957) and Hotelling (1929), in which voters' preferences are assumed to be distributed in some one-dimensional parameter space. Today, the *spatial theory of voting* expresses the idea that voters pick candidates based on the distance in "policy space" between the voter and candidate (Merrill and Grofman 1999). However, this model has generally proven intractable, since the nonconvexity of the politician's objective leads to a difficult (as we have shown) optimization problem, even with a relatively low-dimensional issue space (Roemer 2006). See, Adams et al (2005), Roemer

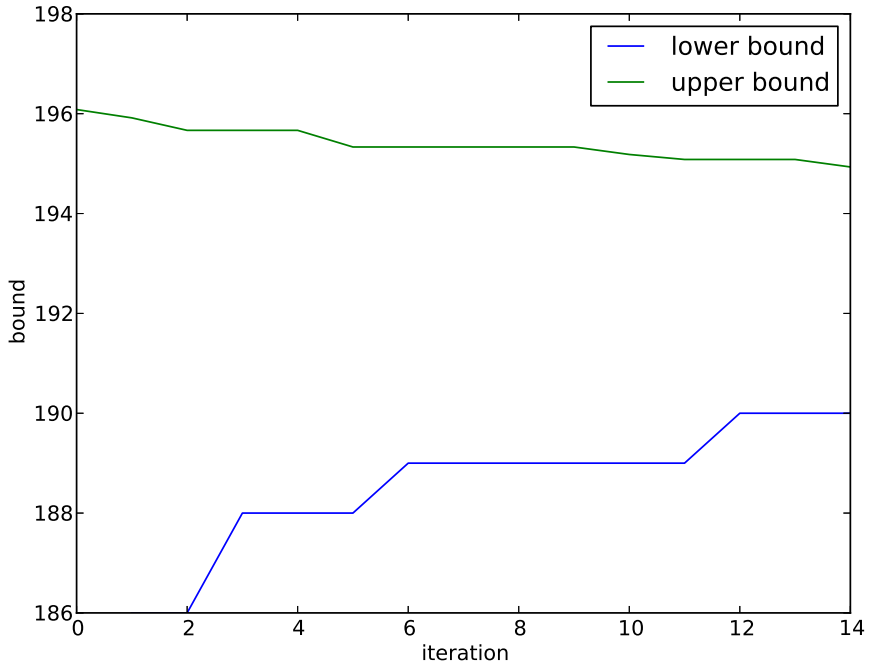

Figure 8: Convergence for NUM with admission function utilities, with $n = 500$ flows over 500 edges, with each flow using on average 2.5 edges each.

(2006, 2004) for more detailed models of party competition based on the spatial theory of voting.

Data. Problem data is generated using responses from the 2008 American National Election Survey (ANES) ANES (2008). Each respondent r in the survey rates each candidate c in the 2008 U.S. presidential election as having positions $y^{rc} \in [1,7]^m$ on m issues. Respondents also say how happy they would be $h^{rc} \in [1,7]$ if the candidate c won. We suppose a respondent would vote for a candidate c if $h^{rc} > h^{rc'}$ for any other candidate c'. If so, $v^{rc} = 1$ and otherwise $v^{rc} = 0$.

For each candidate c and state i, we predict that a respondent $r \in S_i$ in state i will vote for candidate c with probability $\text{logistic}((w_i^c)^T y^{rc})$, depending on the candidate's perceived positions y^{rc}. The parameter vector w_i^c is found by fitting a logistic regression model to the ANES data for each candidate and state pair.

Note that the data from the ANES 2008 survey is not meant to be representative of the population of the US on a state by state basis. It includes information on respondents from only 34 states, some with only 14 respondents.

Optimizing electoral votes. Suppose each state i has p_i votes, which are allocated entirely to the winner of the popular vote. Let $y \in \mathbf{R}^m$ denote the positions the politician takes on each of the m issues. The politician's *pandering* to state i is given by $x_i = w_i^T y$. Using our model, the expected number of votes from state i is

$$f_i(x_i) = p_i \Phi \left(\frac{\text{logistic}(x_i) - .5}{\sqrt{\text{logistic}(x_i)(1 - \text{logistic}(x_i))/N_i}} \right),$$

which is sigmoidal in x_i, where Φ is the normal CDF and N_i is the number of voters in state i. Hence the politician will win the most votes if y is chosen by solving

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & x_i = w_i^T y, \quad i = 1, \ldots, n \\ & 1 \leq y \leq 7. \end{array}$$

Using SP, we find the optimal positions $y^\star$ for Obama (Table 2) to take in the 2008 election, with optimal pandering levels shown in Figure 9. We compare these with average position y_0 that respondents in the survey reported he took. Each graph represents one variable and its associated objective function (solid line) and concave approximation on the rectangle containing the solution (dashed line). The solution for each variable is given by the x coordinates of the red x. The rectangle containing the solution is bounded by the solid grey lines and the endpoints of the interval. The positions are represented using the seven-point Likert scale described in Table 3.

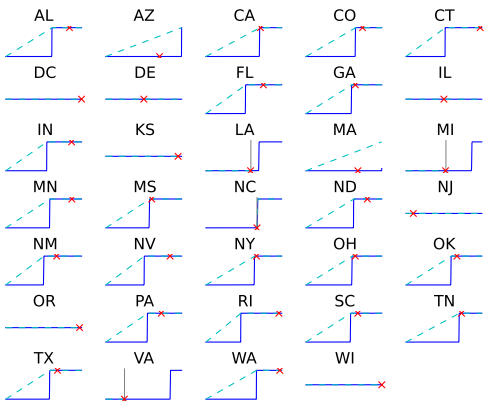

Figure 9: Optimal pandering for Obama in 2008.

A Convergence proof

In this section, we bound the number of convex subproblems that must be computed before obtaining a solution of some specified accuracy to the original sigmoidal programming problem (2).

Table 2: Optimal positions for Obama.

Issue	$y^\star$	y_0
spending/services	1.26	5.30
defense spending	1.27	3.69
liberal/conservative	1.00	3.29
govt assistance to blacks	1.00	3.12

Table 3: Encoding of positions.

Issue	**1**	**7**
spending/services	fewer services	more services
defense spending	decrease spending	increase spending
liberal/conservative	liberal	conservative
govt assistance to blacks	govt should help blacks	blacks should help themselves

Before beginning the proof, we remind the reader of a few of our definitions. For each sigmoidal function $f_i : [l_i, u_i] \to \mathbf{R}$, we define an upper semi-continuous quasi-concave function $h_i : [l_i, u_i] \to \mathbf{R}$ such that

$$f_i(x) = f_i(l_i) + \int_{l_i}^{x} h_i(t)dt, \quad i = 1, \ldots, n.$$

(Such a function h_i always exists.) We let $z_i \in [l_i, u_i]$ be a point maximizing h_i over the interval $[l_i, u_i]$. (This is equivalent to saying that f_i is convex for $x < z_i$, and concave for $x > z_i$.) We define $\hat{f}_i : [l_i, u_i] \to \mathbf{R}$, as before, as the concave envelope of the function f_i, so that $\hat{f}_i(x) \leq f_i(x)$ for every $x \in [l_i, u_i]$. This envelope defines a unique point

$$w_i = \min\{x \in [z_i, u_i] \mid \hat{f}_i(x) = f_i(x)\}.$$

Furthermore, since f_i is sigmoidal,

$$\hat{f}_i(x) = f_i(x), \quad x \in [w_i, u_i].$$

Note in particular that $z_i \leq w_i$.

Recall the operation of the sigmoidal programming algorithm. Let $x^\star = x^\star(Q)$ be the solution to the problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} \hat{f}_i(x_i) \\ \text{subject to} & x \in \mathcal{C} \cap Q. \end{array}$$

So long as the total error is still greater than the desired tolerance ϵ^{des},

$$\sum_{i=1}^{n} \hat{f}_i(x) - f_i(x) \geq \epsilon^{\text{des}},$$

the algorithm proceeds by splitting the rectangle Q in order to attain a better approximation on each of the subrectangles. The rectangle Q is split along the coordinate i with the greatest error $\epsilon_i = \hat{f}_i(x_i^\star) - f_i(x_i^\star)$ into two smaller boxes that meet at $\hat{x}_i^\star$.

We wish to show that this algorithm will terminate after a finite number of splits.

Notice that there can be no error in coordinate i if $x_i^\star \geq w_i$ or if $x_i^\star = l_i$, since $\hat{f}_i$ is the concave envelope of f_i. Furthermore, since f_i and $\hat{f}_i$ are both Lipshitz continuous, the error in the approximation is bounded by a constant (depending on h_i) times the minimum distance of $x_i^\star$ to l_i. We formalize this logic below to show that the size of the rectangles explored cannot be too small, which bounds the number of rectangles we may have to explore.

Error bounds. Since we have chosen h_i to be upper semi-continuous,

$$\begin{aligned}\min_{x\in[l_i,z_i]} h_i(x) &= h_i(l_i),\\ \max_{x\in[l_i,z_i]} h_i(x) &= h_i(z_i).\end{aligned}$$

We use these to bound f_i below on (l_i, z_i):

$$\begin{aligned} f_i(x) &= f_i(l_i) + \int_{l_i}^{x} h_i(t)dt \\ &\geq f_i(l_i) + \left(\min_{s\in(l_i,z_i)} h_i(s)\right)(x - l_i) \\ &\geq f_i(l_i) + h_i(l_i)(x - l_i). \end{aligned}$$

Similarly, we bound $\hat{f}_i$ above on (l_i, z_i):

$$\begin{aligned} \hat{f}_i(x) &= f_i(l_i) + \frac{\int_{l_i}^{w_i} h_i(t)dt}{w_i - l_i}(x - l_i) \\ &\leq f_i(l_i) + \frac{\int_{l_i}^{w_i} \max_{s\in(l_i,z_i)} h_i(s)dt}{w_i - l_i}(x - l_i) \\ &\leq f_i(l_i) + h_i(z_i)(x - l_i). \end{aligned}$$

Hence the error $\epsilon_i(x) = \hat{f}_i(x) - f_i(x)$ in the approximation to the ith objective function f_i at any point $l_i \leq x \leq z_i$ obeys

$$\begin{aligned} \epsilon_i(x) = \hat{f}_i(x) - f_i(x) &= \frac{\int_{l_i}^{w_i} h_i(t)dt}{w_i - l_i}(x - l_i) - \int_{l_i}^{x} h_i(t)dt \\ &\leq (h_i(z_i) - h_i(l_i))\,(x - l_i). \end{aligned}$$

The maximum difference between f_i and $\hat{f}_i$ must occur where f_i is convex, *i.e.*, between l_i and z_i Thus,

$$\begin{aligned}\epsilon_i &= \max_{x\in[l_i,u_i]} \epsilon_i(x) \\ &= \max_{x\in[l_i,z_i]} \epsilon_i(x) \\ &\leq (h_i(z_i) - h_i(l_i))\,(z_i - l_i).\end{aligned}$$

Bounds on rectangle size. A rectangle can be split no further when the error on that rectangle is less than ϵ^{des}. A sufficient condition is $\epsilon_i \leq \epsilon^{\text{des}}/n$ for $i = 1, \ldots, n$. This condition motivates the definition of the minimal side length,

$$\delta_i = \frac{\epsilon^{\text{des}}/n}{h_i(z_i) - h_i(l_i)}.$$

Using equation (3), we see that if each side of the rectangle has

$$\min(z_i, u_i) - l_i \leq \delta_i,$$

then the sufficient condition is met, guaranteeing that the error on the rectangle is less than ϵ^{des}.

Bounds on number of convex subproblems. The worst case for our branch and bound algorithm occurs when the initial rectangle is split into a tiling with smaller rectangles whose side lengths exactly match this minimal size. The maximum number of splits along any dimension is

$$\left\lfloor \frac{z_i - l_i}{\delta_i} \right\rfloor,$$

since a split can only occur if $l_i + \delta_i \leq x_i^\star \leq \min(z_i, u_i)$.

This implies the number of leaves l in the tree of rectangles can be at most

$$l \leq \prod_{i=1}^{n} \left\lfloor \frac{z_i - l_i}{\delta_i} \right\rfloor + 1.$$

A binary tree with l leaves has in total $2l-1$ nodes.Hence the number of convex subproblems solved before the algorithm terminates is bounded by

$$\begin{aligned}2l - 1 &\leq 2\prod_{i=1}^{n} \left\lfloor \frac{z_i - l_i}{\delta_i} \right\rfloor + 1 \\ &= 2\prod_{i=1}^{n} \left\lfloor \frac{(h_i(z_i) - h_i(l_i))\,(z_i - l_i)}{\epsilon^{\text{des}}/n} \right\rfloor + 1.\end{aligned}$$

This shows that the algorithm may require the solution of an exponential number of convex subproblems, in the worst case.

References

Adams JF, Merrill III S, Grofman B (2005) A unified theory of party competition: a cross-national analysis integrating spatial and behavioral factors. Cambridge University Press

Andersen MS, Dahl J, Vandenberghe L (2013) CVXOPT: A Python package for convex optimization, version 1.1.5. URL `abel.ee.ucla.edu/cvxopt`

ANES (2008) The ANES 2008 time series study dataset. `www.electionstudies.org`

Balakrishnan V, Boyd S, Balemi S (1991) Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. International Journal of Robust and Nonlinear Control 1(4):295–317, DOI 10.1002/rnc.4590010404

Balas E (1968) A note on the branch-and-bound principle. Operations Research 16(2):442–445

Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) 2:303–314, 10.1007/BF02551274

Downs A (1957) An economic theory of political action in a democracy. Journal of Political Economy 65(2):135–150

Falk JE, Soland RM (1969) An algorithm for separable nonconvex programming problems. Management Science 15(9):550–569

Fazel M, Chiang M (2005) Network utility maximization with nonconcave utilities using sum-of-squares method. In: 44th IEEE Conference on Decision and Control, pp 1867 – 1874, DOI 10.1109/CDC.2005.1582432

Friedman M, Savage LJ (1948) The utility analysis of choices involving risk. The Journal of Political Economy 56(4):279–304

Henrion D, Lasserre JB (2005) Detecting global optimality and extracting solutions in gloptipoly. In: Henrion D, Garulli A (eds) Positive Polynomials in Control, Lecture Notes in Control and Information Science, vol 312, Springer Berlin Heidelberg, pp 293–310, DOI 10.1007/10997703\X

Horst R, Pardalos PM, Van Thoai N (2000) Introduction to global optimization. Kluwer Academic Pub

Hotelling H (1929) Stability in competition. The Economic Journal 39(153):pp. 41–57

Kahneman D, Tversky A (1979) Prospect theory: An analysis of decision under risk. Econometrica: Journal of the Econometric Society pp 263–291

Karp RM (1972) Reducibility among combinatorial problems. Springer

Lasserre JB (2001) Global optimization with polynomials and the problem of moments. SIAM Journal on Optimization 11:796–817

Lawler EL, Wood DE (1966) Branch-and-bound methods: A survey. Operations Research 14(4):699–719

Makhorin A (2006) GLPK (GNU linear programming kit)

McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. Mathematical programming 10(1):147–175

Merrill S, Grofman B (1999) A Unified Theory of Voting: Directional and Proximity Spatial Models. Cambridge University Press

Parrilo PA (2003) Semidefinite programming relaxations for semialgebraic problems. Mathematical programming 96(2):293–320

Roemer JE (2004) Modeling party competition in general elections. Cowles Foundation Discussion Paper

Roemer JE (2006) Political competition: Theory and applications. Harvard University Press

Sturm JF (1999) Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization methods and software 11(1-4):625–653

Toh KC, Todd MJ, Tütüncü RH (1999) SDPT3 — a MATLAB software package for semidefinite programming, version 1.3. Optimization Methods and Software 11(1-4):545–581

Tversky A, Kahneman D (1992) Advances in prospect theory: Cumulative representation of uncertainty. Journal of Risk and uncertainty 5(4):297–323

Udell M, Boyd S (2014) Bounding duality gap for problems with separable objective, URL `http://www.stanford.edu/~boyd/papers/duality_bound.html`, manuscript.