CENTER FOR
PARALLEL OPTIMIZATION

Bilinear Separation of Two Sets in n-Space

by

Kristin P. Bennett & O. L. Mangasarian

Computer Sciences Technical Report #1109

August 1992

Bilinear Separation of Two Sets in n-Space

Kristin P. Bennett & O. L. Mangasarian *

Abstract

The NP-complete problem of determining whether two disjoint point sets in the n-dimensional real space R^n can be separated by two planes is cast as a bilinear program, that is minimizing the scalar product of two linear functions on a polyhedral set. The bilinear program, which has a vertex solution, is processed by an iterative linear programming algorithm that terminates in a finite number of steps at a point satisfying a necessary optimality condition or at a global minimum. Encouraging computational experience on a number of test problems is reported.

1 Introduction

The problem we wish to consider is the following: Given two disjoint points sets $\mathcal{A}$ and $\mathcal{B}$ in the n-dimensional real space R^n, can they be (strictly) separated by two planes? This is a fundamental NP-complete problem [19, 8] that is depicted in Figure 1 for the 2-dimensional real space R^2. The configurations (a) and (b) of Figure 1 are equivalent as can easily be seen if the roles of $\mathcal{A}$ and $\mathcal{B}$ are interchanged. Bilinear separation is a natural extension of linear separation which, for a long time, has been known to be equivalent to the polynomial-time solution of a single linear program [9, 13, 26, 5]. Linear separation is also equivalent to separation by Rosenblatt's perceptron or linear threshold unit (LTU) [24, 25, 11] (see Figure 2). However most problems are not linearly separable. For example the simple Minsky-Papert exclusive-or classical problem [20], is not linearly separable, but is bilinearly separable. It can be solved by a neural network with 2 layers of linear threshold units [25, 18](see Figure 3).

Other methods of separation by more than one plane, for example multisurface methods (MSM) of pattern separation, have also been proposed [14, 5, 6] and extensively used for medical diagnosis [29, 17, 5]. MSM which has been shown to be equivalent to a feed-forward neural network with a single hidden layer [5], can be trained by a greedy algorithm using linear programming [14, 5, 6]. Thus bilinear and MSM separation can be thought of as alternative linear-programming-based methods for solving problems that are usually solved by neural networks.

It is interesting to note that the bilinear separation depicted in Figure 3(a), which corresponds to the topology of the bilinear separation of Figure 1(a) and 1(b), can be represented by a single hidden layer neural network with two hidden units and one output unit. However this is not the case for a bilinear separation with the topology of Figure 1(c). In fact if we separated the exclusive-or example by planes using this topology, as shown in Figure 4(a), the corresponding neural net depicted in Figure 4(b), requires two hidden layers each with 2 units and one output unit. This indicates that this is a more complex separation from a neural network point of view . It will turn

*Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, Wisconsin 53706. Email bennett@cs.wisc.edu, olvi@cs.wisc.edu. This material is based on research supported by Air Force Office of Scientific Research Grant AFOSR-89-0410, National Science Foundation Grant CCR-9101801, and Air Force Laboratory Graduate Fellowship SSAN 531-56-2969.

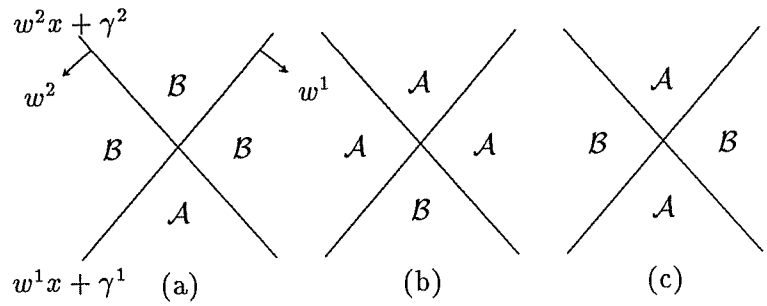
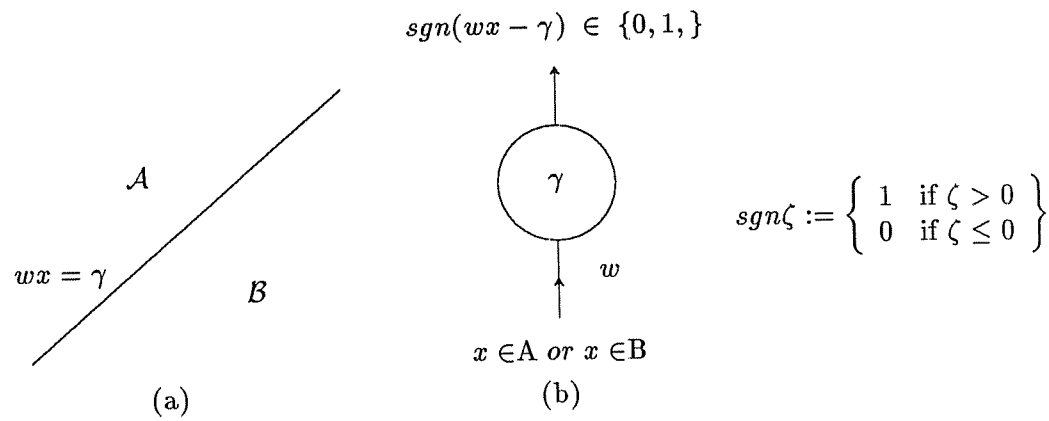
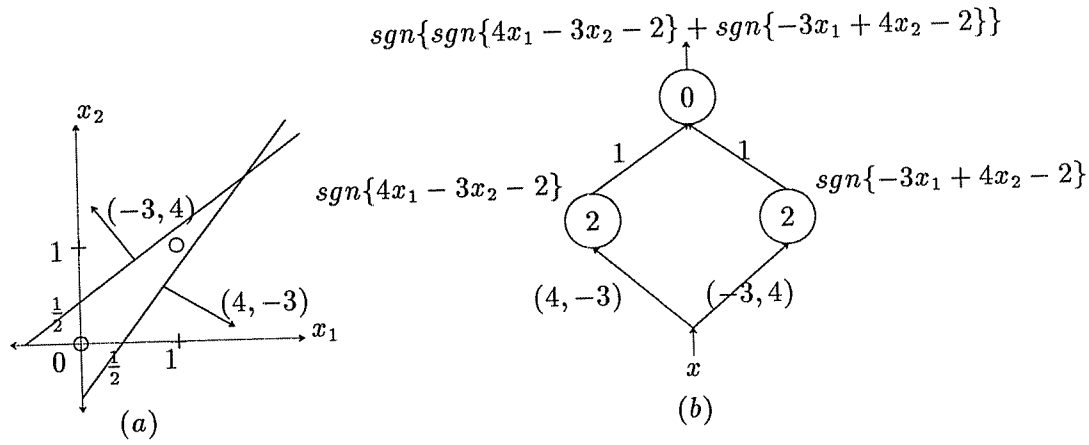


Figure 1: Bilinearly Separable Sets $\mathcal{A}$ and $\mathcal{B}$ in R^n



(*a*) *Linearly separable sets*
(*b*) *Equivalent linear threshold unit with incoming weights* w *and threshold* γ

Figure 2: Linear Separator and Equivalent Linear Threshold Unit



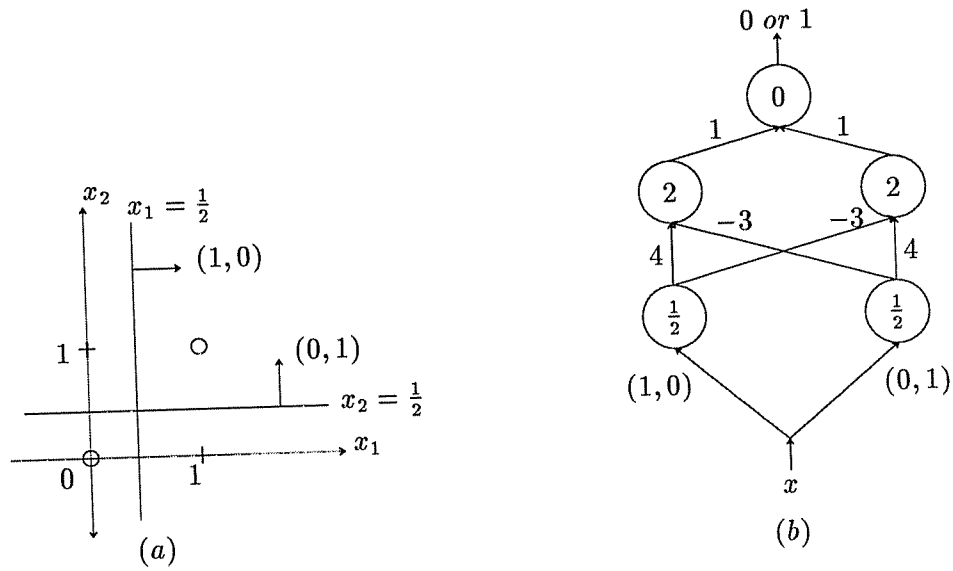
(a) *Bilinearly separable exclusive – or example* : $\mathcal{A} = \{(0,0),(1,1)\}$, $\mathcal{B} = \{(1,0),(0,1)\}$
(b) *Equivalent neural network*

Figure 3: Bilinear Separator for Exclusive-Or Example and Equivalent Neural Network

out that this separation leads also to a more difficult bilinear program with coupled constraints: program (15) instead of program (13) below.

Our objective in this paper is to present an equivalent mathematical programming formulation to the bilinear separation problem, which turns out to be a bilinear program on a polyhedral set (Theorem 3.1). We shall solve this bilinear program by a finite sequence of linear programs (Algorithms 2.1, 2.2 and 2.3) which terminate either at a solution of the bilinear program or at a point satisfying the minimum principle necessary optimality conditions [15, Theorem 9.3.3]. In Section 2 we begin with some basic results on the existence of vertex solutions to bilinear programs as well as some linear-programming-based finite algorithms for their solution. Because of the special property of a zero minimum for bilinearly separable problems, we have opted for the simpler Franke-Wolfe type algorithms [10], rather than the more complex algorithms that have been given for bilinear programs [1, 30, 27, 28]. Our computational experience, summarized in Section 4, indicates that the proposed algorithms are effective ones, especially in view of the fact that the underlying problem is an NP-complete problem. (More precisely only the bilinear separability problem corresponding to Figure 1(a)-1(b) has been shown to be NP-complete [19, 8]. That the corresponding problem to Figure 1(c) is NP-complete as well, can be deduced from Theorem 3.1 below by noting that the bilinear program (13) is a special instance of the bilinear program (15).)

A word about our notation now. For a vector x in the n-dimensional real space R^n, x_+ will denote the vector in R^n with components $(x_+)_i := \max\{x_i, 0\}$, $i = 1,\ldots,n$. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix, A' will denote the transpose while A_i will denote the ith row. A vector of ones in a real space of arbitrary dimension will be denoted by e.



(*a*) *Bilinear separation using the topology of Figure* 1(*c*)
(*b*) *Equivalent neural network*

Figure 4: Alternative Bilinear Separator for Exclusive-Or Example and Equivalent Neural Network

2 Vertex Solutions and Finite Algorithms for Bilinear Programs

We present here some simple basic results for bilinear programs which show under what conditions these problems have vertex solutions. These results are then used to generate finite linear-programming-based algorithms for solving bilinear separability problems as bilinear programs (see Section 3). We shall consider two categories of bilinear programs corresponding to cases (a)-(b) and case (c) of Figure 1. The first case will have uncoupled constraints while the second case will have coupled constraints. We note that there are many papers on bilinear programs such as [27, 28] with uncoupled constraints, and [1, 30] with coupled constraints. However, none of the papers on coupled constraints appear to exploit zeroness of the minimum as we do here for the case of bilinearly separable sets. This fact allows us to conclude that a vertex solution exists for such problems. This in turn leads to finite termination for the proposed algorithms. For uncoupled constraints, the existence of a vertex solution is known and has been exploited algorithmically [27, 28]. For completeness and contrast with the proof for the case of coupled bilinear programs, we begin by a simple proof of the existence of a vertex solution for uncoupled bilinear programs.

Proposition 2.1 (Existence of vertex solutions for uncoupled bilinear programs) *If the uncoupled bilinear program*

$$\min_{x,y,r,s} \{ xy \mid Cx + Er \geq g,\ Dy + Fs \geq h,\ (x,y,r,s) \geq 0\} \tag{1}$$

is feasible, then it has a vertex solution.

Proof. Since the quadratic objective function is bounded below by zero on the polyhedral feasible region, it must have a solution $(\bar{x}, \bar{y}, \bar{r}, \bar{s})$ [10]. Hence the linear program

$$\min_{(x,r)\in\mathcal{X}} x\bar{y},\ \ \mathcal{X} = \{ Cx + Er \geq g,\ (x,r) \geq 0\}$$

has a vertex $(\hat{x}, \hat{r})$ of its feasible region $\mathcal{X}$ as solution [22], and such that

$$\hat{x}\bar{y} = \bar{x}\bar{y}.$$

Similarly the linear program

$$\min_{(y,s)\in\mathcal{Y}} \hat{x}y,\ \ \mathcal{Y} = \{ Dy + Fs \geq h,\ (y,s) \geq 0\}$$

has a vertex $(\hat{y}, \hat{s})$ of its feasible region $\mathcal{Y}$ as solution, and such that

$$\hat{x}\hat{y} = \hat{x}\bar{y} = \bar{x}\bar{y}.$$

Hence $((\hat{x}, \hat{r}), (\hat{y}, \hat{s}))$ is a vertex of $\mathcal{X} \times \mathcal{Y}$ and a vertex solution of (1). □

To establish the existence of a vertex solution to a coupled bilinear program we require the additional assumption that the minimum value of the bilinear objective be zero. This does not affect the application to bilinear separation, since the objective function does indeed become zero for bilinearly separable sets.

Proposition 2.2 (Existence of vertex solution for coupled bilinear programs) *If the coupled bilinear program*

$$\min_{x,y,r} \{ xy \mid Cx + Dy + Er \geq g,\ (x,y,r) \geq 0\} \tag{2}$$

has a zero minimum, then it has a vertex solution.

Proof. Let $\mathcal{S}$ denote the feasible region of (2). Note first that

$$xy = 0,\ (x,y) \geq 0 \ \Leftrightarrow \ x - (x-y)_+ = 0$$

and for $x, y \in R^n$ and $i = 1, \ldots, n$

$$(x,y,r) \in \mathcal{S} \ \Rightarrow \ (x,y) \geq 0$$
$$\Rightarrow \ ((x-(x-y)_+)_i = \left\langle \begin{array}{ll} y_i & \text{if } x_i - y_i \geq 0 \\ x_i & \text{if } x_i - y_i < 0 \end{array} \right\rangle \geq 0$$

Hence

$$0 = \min_{(x,y,r)\in\mathcal{S}} xy \ \Leftrightarrow \ 0 = \min_{(x,y,r)\in\mathcal{S}} e(x-(x-y)_+)$$

However $e(x-(x-y)_+)$ is a concave function because $e(x-y)_+$ is convex. Since we are minimizing a concave function on a polyhedral set not containing lines going to infinity in both directions, it must have a vertex solution [23, Corollary 32.3.4]. □

The above proof is based on the proofs of Lemmas 1 and 2 of [16] which show that every solvable linear complementarity problem, monotonic or not, has a vertex solution. With the above theorems we can formulate finite algorithms for each of the uncoupled and coupled bilinear programs: algorithms UBPA 2.1, UBPA1 2.2 and BPA 2.3.

Algorithm 2.1 (Uncoupled bilinear program algorithm (UBPA)) *Start with any feasible point* (x^0, y^0, r^0, s^0) *for (1). Determine* $(x^{i+1}, y^{i+1}, r^{i+1}, s^{i+1})$ *from* (x^i, y^i, r^i, s^i) *as follows:*

$$(x^{i+1}, r^{i+1}) \in \textit{ arg vertex partial } \min_{x,r} \{\, xy^i \mid Cx + Er \geq g,\ (x,r) \geq 0\}$$

$$(y^{i+1}, s^{i+1}) \in \textit{ arg vertex partial } \min_{y,s} \{\, x^{i+1}y \mid Dy + Fs \geq h,\ (y,s) \geq 0\}$$

***and** such that* $x^{i+1}y^{i+1} < x^iy^i$. *Stop when impossible.*

In the above algorithm, "*arg vertex partial* min" denotes a vertex in the solution set of the indicated linear program, or any vertex along the path of the simplex or other pivotal method that attempts to decrease the objective function. The combined decrease of both steps must be such that $x^{i+1}y^{i+1} < x^iy^i$. We establish now finiteness of the above algorithm.

Theorem 2.1 (Finite termination of UBPA) *Let the uncoupled bilinear program (1) be feasible. Then UBPA 2.1 terminates in a finite number of steps at a global solution or a point* $(x^{i+1}, y^i, r^{i+1}, s^i)$ *that satisfies the minimum principle necessary optimality condition [15]*

$$y^i(x - x^{i+1}) + x^{i+1}(y - y^i) \geq 0 \quad \forall\, (x,r) \in \mathcal{X},\ \forall\, (y,s) \in \mathcal{Y}. \tag{3}$$

Proof. If for some i, $x^{i+1}y^{i+1} \not< x^iy^i$, then each of the linear programs of UBPA must have been solved to optimality and

$$xy^i \geq x^iy^i = x^{i+1}y^i = x^{i+1}y^{i+1} \leq x^{i+1}y, \ \forall\, (x,r) \in \mathcal{X},\ \forall\, (y,s) \in \mathcal{Y}$$

from which the minimum principle condition (3) follows. Since there are a finite number of vertices of $\mathcal{X} \times \mathcal{Y}$, and since each vertex visited by UBPA is less than the previous one in xy-value, no vertex is repeated. Thus UBPA must terminate at either a global minimum or a point satisfying the minimum principle (3). □

Note that UBPA is serial in nature in that (y^{i+1}, s^{i+1}) is computed **after** (x^{i+1}, r^{i+1}) is computed. A parallel version where both (x^{i+1}, r^{i+1}) and (y^{i+1}, s^{i+1}) can be computed simultaneously, can be shown to possess the finite termination property. We give it below and skip its proof which is completely analogous to that of Theorem 2.1.

Algorithm 2.2 (Uncoupled bilinear program algorithm 1 (UBPA1)) *Start with any feasible point* (x^0, y^0, r^0, s^0) *for (1). Determine* $(x^{i+1}, y^{i+1}, r^{i+1}, s^{i+1})$ *from* (x^i, y^i, r^i, s^i) *as follows:*

$$(\tilde{x}^{i+1}, \tilde{r}^{i+1}) \in \textit{arg vertex partial} \min_{x,r} \{\, xy^i \mid Cx + Er \geq g,\ (x,r) \geq 0\}$$

$$(\tilde{y}^{i+1}, \tilde{s}^{i+1}) \in \textit{arg vertex partial} \min_{y,s} \{\, x^i y \mid Dy + Fs \geq h,\ (y,s) \geq 0\}$$

$$(x^{i+1}, y^{i+1}) \in \{(\tilde{x}^{i+1}, y^i), (x^i, \tilde{y}^{i+1}), (\tilde{x}^{i+1}, \tilde{y}^{i+1})\}$$

and *such that* $x^{i+1}y^{i+1} < x^i y^i$. *Stop when impossible.*

We turn our attention now to the more general case and give a finite Frank-Wolfe algorithm for the coupled bilinear program (2) when its objective function has a zero minimum.

Algorithm 2.3 (Bilinear program algorithm (BPA)) *Start with any feasible point* (x^0, y^0, r^0) *for (2). Determine* $(x^{i+1}, y^{i+1}, r^{i+1})$ *from* (x^i, y^i, r^i) *as follows:*

- $$(u^i, v^i, w^i) \in \textit{arg vertex} \min_{(x,y,r)} \{\, xy^i + x^i y \mid Cx + Dy + Er \geq g, (x,y,r) \geq 0\}$$
- $$\textit{Stop if } u^i y^i + x^i v^i = 2x^i y^i$$
- $$(x^{i+1}, y^{i+1}, r^{i+1}) = (1-\lambda^i)(x^i, y^i, r^i) + \lambda^i (u^i, v^i, w^i) \textit{ where}$$
$$\lambda^i \in \textit{arg} \min_{0 \leq \lambda \leq 1} (x^i + \lambda(u^i - x^i))(y^i + \lambda(u^i - y^i))$$

Theorem 2.2 (Convergence and finite termination theorem for BPA) *Either the sequence* $\{(x^i, y^i, r^i)\}$ *of BPA terminates at some* $\{(x^j, y^j, r^j)\}$ *satisfying the minimum principle necessary optimality condition*

$$y^j(x - x^j) + x^j(y - y^j) \geq 0 \quad \forall\, (x,y,r) \in \mathcal{S}, \tag{4}$$

where $\mathcal{S}$ *is the feasible region of (2), or each accumulation point* $(\bar{x}, \bar{y}, \bar{z})$ *of* $\{(x^i, y^i, r^i)\}$ *satisfies the minimum principle. If* $\bar{x}\bar{y} = 0$, *then one of the vertices of the sequence* $\{(u^i, v^i, w^i)\}$ *solves the bilinear program (2) and* $u^i v^i = 0$.

Proof. Follows from Theorems A.1 and A.2 of the Appendix. □

3 Equivalence of Bilinear Separability and Bilinear Programming

In this part of the paper we will show how to reduce the bilinear separability problem to one of three bilinear programs. We consider the disjoint point sets $\mathcal{A}$ and $\mathcal{B}$ in R^n made up of m and k points and represented by the matrices $A \in R^{m \times n}$ and $B \in R^{k \times n}$ respectively. When the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ are also disjoint then $\mathcal{A}$ and $\mathcal{B}$ are said to be **linearly separable**. By using the duality theory of linear programming, this can be shown to be equivalent to the existence of a plane $wx = \gamma$ strictly separating $\mathcal{A}$ from $\mathcal{B}$ (see Figure 2(a)) which is equivalent to [12, 13, 26, 5]

$$-Aw + e\gamma + e \leq 0, \quad Bw - e\gamma + e \leq 0, \quad \text{for some } w \in R^n,\ \gamma \in R. \tag{5}$$

Based on the above definition of linear separability, we now define bilinear separability as follows:

Definition 3.1 (Bilinear separability definition (See Figure 5)) *The sets $\mathcal{A}$ and $\mathcal{B}$ are bilinearly separable if and only at least one of the following systems of disjunctive linear inequalities is solvable for $(w^1, w^2, \gamma^1, \gamma^2)$ thus giving the separating planes $w^1 x = \gamma^1$ and $w^2 x = \gamma^2$:*

$$\left\langle \begin{array}{lll} -Aw^1 + \gamma^1 e + e \leq 0 & \textbf{and} & -Aw^2 + \gamma^2 e + e \leq 0 \\ B_i w^1 - \gamma^1 + 1 \leq 0 & \textbf{or} & B_i w^2 - \gamma^2 + 1 \leq 0,\ i = 1, \ldots, k \end{array} \right\rangle \tag{6}$$

$$\left\langle \begin{array}{lll} -Bw^1 + \gamma^1 e + e \leq 0 & \textbf{and} & -Bw^2 + \gamma^2 e + e \leq 0 \\ A_i w^1 - \gamma^1 + 1 \leq 0 & \textbf{or} & A_i w^2 - \gamma^2 + 1 \leq 0,\ i = 1, \ldots, m \end{array} \right\rangle \tag{7}$$

$$\left\langle \begin{array}{c} \left\langle \begin{array}{c} (-A_i w^1 + \gamma^1 + 1 \leq 0 \ \textbf{and} \ -A_i w^2 + \gamma^2 + 1 \leq 0) \\ \textbf{or} \\ (A_i w^2 - \gamma^2 + 1 \leq 0 \ \textbf{and} \ A_i w^1 - \gamma^1 + 1 \leq 0),\ i = 1, \ldots, m \end{array} \right\rangle \\ \\ \left\langle \begin{array}{c} (B_i w^1 - \gamma^1 + 1 \leq 0 \ \textbf{and} \ -B_i w^2 + \gamma^2 + 1 \leq 0) \\ \textbf{or} \\ (B_i w^2 - \gamma^2 + 1 \leq 0 \ \textbf{and} \ -B_i w^1 + \gamma^1 + 1 \leq 0),\ i = 1, \ldots, k \end{array} \right\rangle \end{array} \right\rangle \tag{8}$$

It is easy to see that the above definition can be stated in the following alternative form.

Definition 3.2 (Alternative bilinear separability definition) *The sets $\mathcal{A}$ and $\mathcal{B}$ are bilinearly separable if and only if at least one of the following systems of inequalities is solvable for $(w^1, w^2, \gamma^1, \gamma^2)$:*

$$\begin{array}{l} -Aw^1 + \gamma^1 e + e \leq 0 \ and \ -Aw^2 + \gamma^2 e + e \leq 0 \\ (Bw^1 - \gamma^1 e + e)_+ (Bw^2 - \gamma^2 e + e)_+ \ = \ 0 \end{array} \tag{9}$$

$$\begin{array}{l} -Bw^1 + \gamma^1 e + e \leq 0 \ and \ -Bw^2 + \gamma^2 e + e \leq 0 \\ (Aw^1 - \gamma^1 e + e)_+ (Aw^2 - \gamma^2 e + e)_+ \ = \ 0 \end{array} \tag{10}$$

$$\begin{array}{l} ((-Aw^1 + \gamma^1 e + e)_+ + (-Aw^2 + \gamma^2 e + e)_+)((Aw^2 - \gamma^2 e + e)_+ + (Aw^1 - \gamma^1 e + e)_+)) \ = \ 0 \\ ((Bw^1 - \gamma^1 e + e)_+ + (-Bw^2 + \gamma^2 e + e)_+)((Bw^2 - \gamma^2 e + e)_+ + (-Bw^1 + \gamma^1 e + e)_+)) \ = \ 0 \end{array} \tag{11}$$

To reduce the above definition of bilinear separability to a bilinear program, we make use of the following simple lemma.

Lemma 3.1 *Let $T^i \subset R^m$, $i = 1, \ldots, 4$. Then*

$$\left\langle \begin{array}{l} t^i \in T^i,\ i = 1, \ldots, 4 \\ (t^1_+ + t^2_+)(t^4_+ + t^3_+) = 0 \end{array} \right\rangle \Leftrightarrow \left\langle \begin{array}{lll} 0 \ = & \min\limits_{y^i, t^i, i=1,\ldots,4} & (y^1 + y^2)(y^4 + y^3) \\ such\ that & & t^i \in T^i,\ t^i \leq y^i,\ 0 \leq y^i \\ & & i = 1, \ldots, 4 \end{array} \right\rangle \tag{12}$$

Proof. ($\Rightarrow$) Let $\hat{t}^i$, $i = 1, \ldots, 4$ solve the first problem. Define $\hat{y}^i := \hat{t}^i_+ \geq \hat{t}^i$, $i = 1, \ldots, 4$. Then $\hat{y}^i \geq \hat{t}^i$, $\hat{y}^i \geq 0$, $\hat{t}^i \in T^i$, for $i = 1, \ldots, 4$. Hence $(\hat{y}^i, \hat{t}^i)$, $i = 1, \ldots, 4$, is feasible for the second problem, and is optimal because

$$(\hat{y}^1_+ + \hat{y}^2_+)(\hat{y}^4_+ + \hat{y}^3_+) \ = \ (\hat{t}^1_+ + \hat{t}^2_+)(\hat{t}^4_+ + \hat{t}^3_+) \ = \ 0.$$

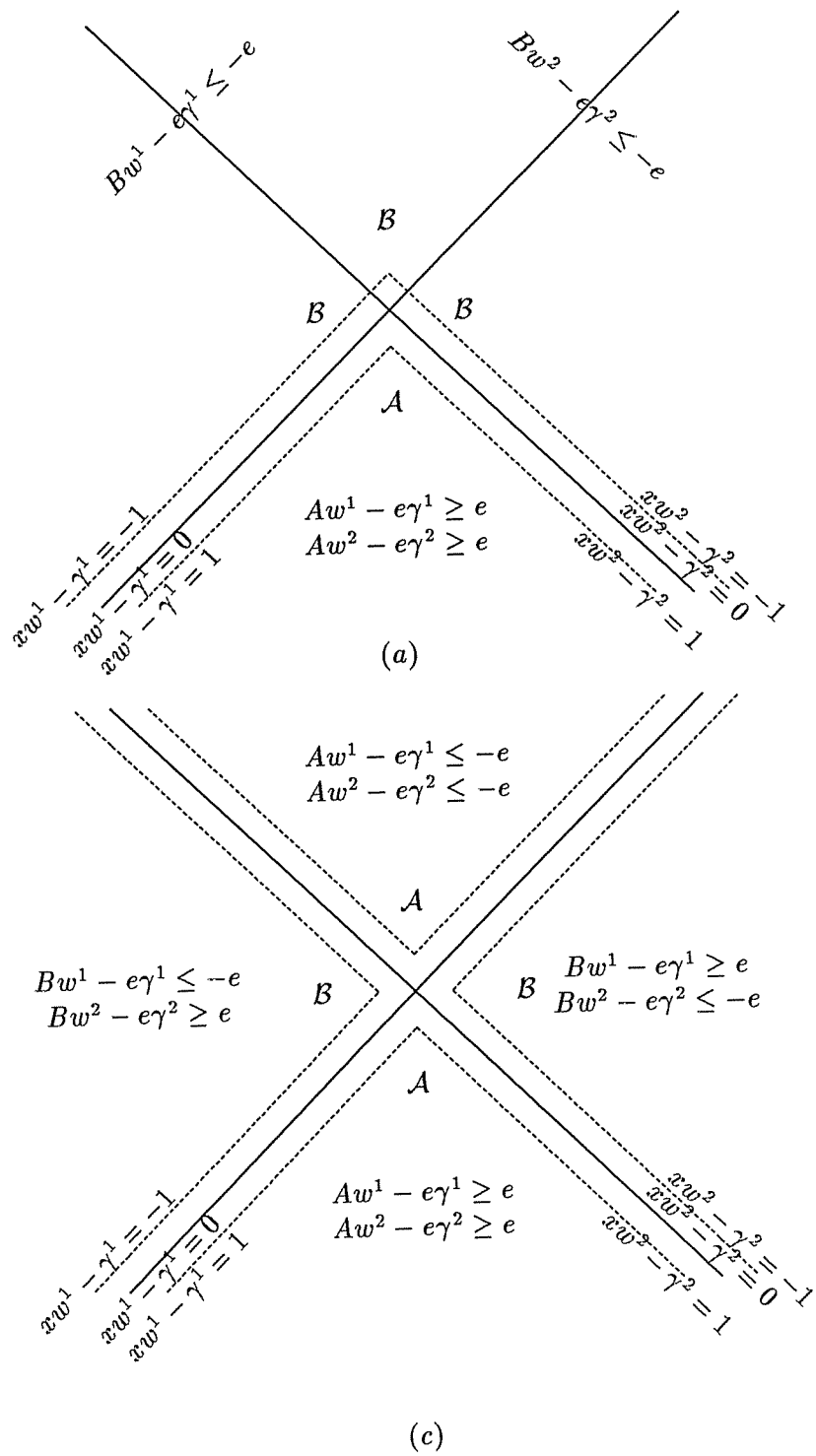


Figure 5: Geometric Representation of Bilinear Separability Definition

($\Leftarrow$) Let $(\hat{y}^i, \hat{t}^i)$, $i = 1, \ldots, 4$, solve the second problem. Then

$$(\hat{y}_j^1 + \hat{y}_j^2) = 0 \text{ or } (\hat{y}_j^4 + \hat{y}_j^3) = 0,\ j = 1, \ldots, m$$

Consequently

$$(\hat{t}_j^1 \leq 0 \text{ and } \hat{t}_j^2 \leq 0) \text{ or } (\hat{t}_j^4 \leq 0 \text{ and } \hat{t}_j^3 \leq 0),\ j = 1, \ldots, m$$

Hence

$$(\hat{t}_{j+}^1 + \hat{t}_{j+}^2) = 0 \text{ or } (\hat{t}_{j+}^4 + \hat{t}_{j+}^3) = 0,\ j = 1, \ldots, m$$

and

$$(\hat{t}_+^1 + \hat{t}_+^2)(\hat{t}_+^4 + \hat{t}_+^3) \ = \ 0.$$

□

We now formulate the alternate bilinear separability definition as a bilinear program with the help of this lemma.

Theorem 3.1 (Equivalence of bilinear separability to bilinear programming) *The sets $\mathcal{A}$ and $\mathcal{B}$ are bilinearly separable if and only if at least one of the following bilinear programs has a zero minimum in which case a minimum solution $(\hat{w}^1, \hat{w}^2, \hat{\gamma}^1, \hat{\gamma}^2)$ determines the separating planes: $\hat{w}^1 x = \hat{\gamma}^1$ and $\hat{w}^2 x = \hat{\gamma}^2$:*

$$\begin{array}{ll} \textit{minimize} & z^1 z^2 \\ z^1, z^2, w^1, w^2, \gamma^1, \gamma^2 & \\ \textit{such that} & -Aw^1 + \gamma^1 e + e \leq 0, \quad -Aw^2 + \gamma^2 e + e \leq 0 \\ & Bw^1 - \gamma^1 e + e \leq z^1, \quad Bw^2 - \gamma^2 e + e \leq z^2 \\ & 0 \leq z^1 \qquad\qquad 0 \leq z^2 \end{array} \tag{13}$$

$$\begin{array}{ll} \textit{minimize} & y^1 y^2 \\ y^1, y^2, w^1, w^2, \gamma^1, \gamma^2 & \\ \textit{such that} & -Bw^1 + \gamma^1 e + e \leq 0, \quad -Bw^2 + \gamma^2 e + e \leq 0 \\ & Aw^1 - \gamma^1 e + e \leq y^1, \quad Aw^2 - \gamma^2 e + e \leq y^2 \\ & 0 \leq y^1 \qquad\qquad 0 \leq y^2 \end{array} \tag{14}$$

$$\begin{array}{ll} \textit{minimize} & (y^1 + y^2)(y^4 + y^3) \ + \ (z^1 + z^2)(z^4 + z^3) \\ y^1, y^2, y^3, y^4, z^1, z^2, z^3, z^4, & \\ w^1, w^2, \gamma^1, \gamma^2 & \\ \textit{such that} & -Aw^1 + \gamma^1 e + e \leq y^1, \quad -Aw^2 + \gamma^2 e + e \leq y^2 \\ & 0 \leq y^1 \qquad\qquad 0 \leq y^2 \\ & Aw^1 - \gamma^1 e + e \leq y^3, \quad Aw^2 - \gamma^2 e + e \leq y^4 \\ & 0 \leq y^3 \qquad\qquad 0 \leq y^4 \\ & Bw^1 - \gamma^1 e + e \leq z^1, \quad -Bw^2 + \gamma^2 e + e \leq z^2 \\ & 0 \leq z^1 \qquad\qquad 0 \leq z^2 \\ & -Bw^1 + \gamma^1 e + e \leq z^3, \quad Bw^2 - \gamma^2 e + e \leq z^4 \\ & 0 \leq z^3 \qquad\qquad 0 \leq z^4 \end{array} \tag{15}$$

Proof. Use Lemma 3.1 on Definition 3.2. □

Remark 3.1 *Note that there is a key difference between problems (13)-(14) and (15) above. The former have uncoupled constraints in (z^1, w^1, γ^1) and (z^2, w^2, γ^2), and in (y^1, w^1, γ^1) and (y^2, w^2, γ^2), while (15) does not. This allows us to use the simplex method and the finitely terminating UBPA 2.1 and UBPA1 2.2 on (13)-(14).*

Remark 3.2 *In order to apply the results of Section 2, all the variables in the above bilinear programs need to be nonnegative. This can be easily done by adding two nonnegative one-dimensional variables* ζ^1, ζ^2 *to the problem and defining* $w^1 = \hat{w}^1 - e\zeta^1$, $w^2 = \hat{w}^2 - e\zeta^2$, $\gamma^1 = \hat{\gamma}^1 - \zeta^1$, $\gamma^2 = \hat{\gamma}^2 - \zeta^2$, $(\hat{w}^1, \hat{w}^2, \hat{\gamma}^1, \hat{\gamma}^2, \zeta^1, \zeta^2) \geq 0$.

4 Computational Results

Both the uncoupled and coupled bilinear programming algorithms were implemented and tested on a suite of problems. The linear programming package MINOS 5.4 [21] was used to solve the linear subproblems. Initial experimentation with UBPA 2.1, UBPA1 2.2, and BPA 2.3 showed that they were prone to halting at solutions with $w^1 = 0$, $w^2 = 0$, or $w^1 = w^2$ which satisfied the minimum principle necessary optimality condition. These are clearly undesirable solutions that are easily detected in practice. The $w^1 = w^2$ case for the uncoupled problem was practically avoided by starting with a good initial solution found by using the first two planes of the Multisurface-Method-Tree algorithm [4, 6]. The problem of $w^1 = 0$ and $w^2 = 0$ was tackled by detecting when this occurred and then adding a linear constraint to the problem which made the zero solution infeasible. This did not affect the convergence proof of the algorithm. Care must be taken to avoid excluding the optimal solution.

We tested the algorithms on two-dimensional toy problems and on randomly generated problems in high dimensions. Figure 6 shows a solution found by the UBPA1 2.2 on a two-dimensional problem with 65 points in $\mathcal{A}$ and 102 points $\mathcal{B}$. The algorithm was able to completely separate the two sets in 157 simplex pivots. The bilinear separation found by BPA 2.3 in 829 simplex pivots on the other bilinearly separable problem in two dimensions with 130 points in $\mathcal{A}$ and 74 points $\mathcal{B}$ is depicted in Figure 7.

In order to investigate the effectiveness of the linear programming approach on high-dimensional bilinearly separable problems, we randomly generated test problems as follows [2]. Two points were randomly generated on an n-dimensional unit sphere. These points were used as the normals w^1 and w^2 for the two separating planes $w^1x = 0$ and $w^2x = 0$ with the topology of either of Figure 1(a) or Figure 1(c). Then a training set of l randomly generated points on the n-dimensional unit sphere was generated and classified using the two planes. Similarly a testing set of consisting of 5000 points was also generated. The bilinear algorithm appropriate for the topology was run using the points in the training set, and then used to classify the points in the testing set. For the uncoupled problems, we performed these experiments for 10, 25, 50, and 100 dimensional problems using 500 and 1000 training points. For the coupled problems, we performed these experiments for 5, 10, and 25 dimensional problems using 500 and 1000 training points. These results are summarized in Tables 1 and 2 and in Figures 8 and 9.

Table 1 summarizes the training set performance on the uncoupled problems using UBPA 2.1. Table 2 summarizes the training set performance on the coupled problems using BPA 2.3. The algorithms correctly determined that the sets were bilinearly separable in every single case. We should note that we used the knowledge that $\gamma^1 = \gamma^2 = 0$ at the solution to constrain the algorithm when $w^1 = 0$ or $w^2 = 0$ was encountered. However, even with $\gamma^1 = \gamma^2 = 0$, the underlying problem is still NP-complete.

Figure 8 depicts the performance of the bilinear separation on an unseen testing set made up of 5000 points. The curves in this figure (as well as those in Figure 9) are the average of 10 runs, and are marked with 95% confidence intervals. Curves for two bilinear separations are given: one trained on a 500-point set and one on a 1000-point set. These curves indicate that the trained bilinear separation is able to learn the underlying structure quite accurately for small dimensional problems,

as seen from the small errors for these problems. But as the problem dimension increases, the error increases. However if more training examples are used the error decreases. This is indicated by lower curves in Figure 8 for the 1000-point training set compared to the curves for the 500-point training set. These trends agree with computational learning theory results [3] that provide necessary and sufficient conditions for valid generalization (that is correctness on testing sets) which are dependent on problem dimension and number of points in the training set.

To measure how well the algorithms scale as the dimension grows we examined the average number of total simplex pivots to solve each problem. These results are reported in Figure 9 for the uncoupled and coupled algorithms UBPA 2.1 and BPA 2.3. The computational cost of the coupled algorithm is considerably higher than that of the uncoupled algorithm. Not only does the coupled algorithm require more pivots, but each pivot takes longer since the number of rows in problem (15) to which the coupled algorithm BPA 2.3 is applied is twice as many as the number of rows in problem (13) to which the uncoupled algorithm UBPA 2.1 is applied. Also the number of columns in problem (15) is approximately four times as many as the number of columns in problem (13). Thus the more demanding problems solved by the coupled algorithm BPA 2.3 were of smaller maximum dimension, 25 instead of 100, than for the problems solved by the uncoupled algorithm UBPA 2.1. We did however successfully solve with the coupled algorithm, problems with as many 1000 points in 100 dimensional space, but did not run 10 such cases to average and report. Note that in Figure 9 the number of pivots required for the uncoupled algorithm UBPA 2.1 for a training set size of 500 in 100 dimensions is considerably lower than the corresponding number for the 50-dimensional case. This is because 500 is not a sufficient number of training set examples to adequately represent a problem in a 100-dimensional space. Nine out of ten generated training sets were linearly separable, and were quickly separated linearly by the Multisurface-Method-Tree algorithm [4, 6].

We also compared the performance of the UBPA 2.1 and BPA 2.3 with that of the backpropagation (BP) algorithm [25], the standard algorithm for training neural networks. BP was used to train neural networks configured as in Figure 3(b) and Figure 4(b) to solve the bilinear separability problem. We found that BP could consistently solve only small dimensional problems with the topology of Figure 1(a). Specifically BP completely separated five out of five such 5-dimensional problems. However BP solved only one out of five problems in 5-dimensional space with the topology of Figure 1(c). Also BP failed to solve higher dimensional problems for both topologies within 40,000 epochs. Unlike UBPA 2.1 and BPA 2.3, BP has no well defined stopping criterion and hence may have failed because it was not given sufficient processing time. We found BP to be computationally more costly than the bilinear programming approach for all but small problems. For example, in the 10-dimensional case with the the topology of Figure 1(c) and 500 training examples, BP failed after an average of 8204 seconds, while BPA correctly solved this problem in 691 seconds on average.

5 Conclusion

The NP-complete bilinear separation problem was posed as one of two bilinear programs. These bilinear programs were shown to have vertex solutions for bilinearly separable problems. This property led to linear programming based algorithms with finite termination property. Computational experiments indicated the viability of these algorithms for problems of up to 100 dimensions and 1000 points. An important computational achievement to point out is the fact that all the 140 cases of the NP-complete problems, represented in Tables 1 and 2, were all solved correctly without a single failure.

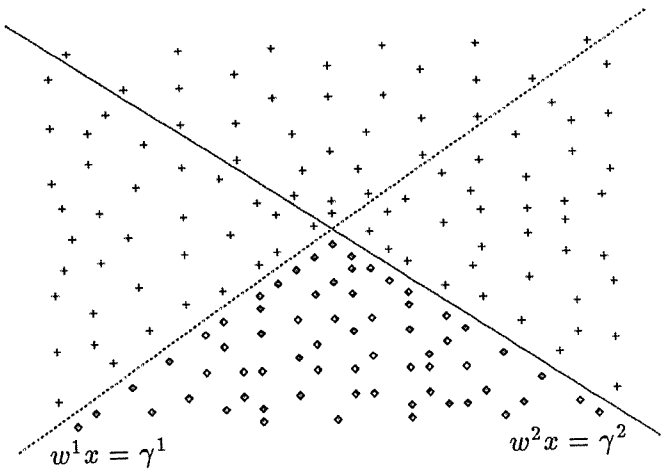


Figure 6: Bilinear Separation Found by UBPA1

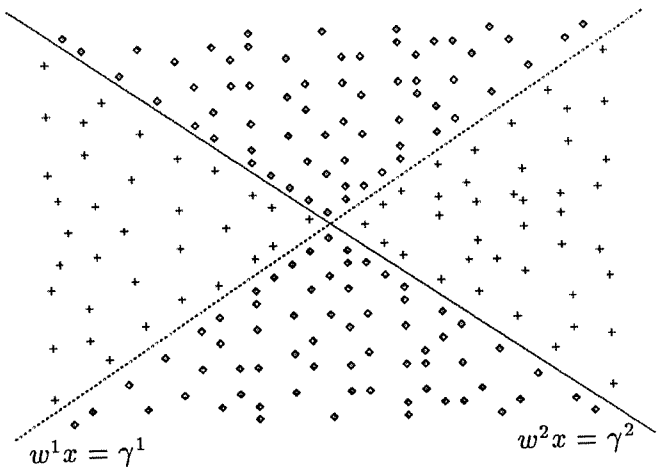


Figure 7: Bilinear Separation Found by BPA

Dimension n	Training Set Size	100% Bilinear Separation
10	500	10 of 10
10	1000	10 of 10
25	500	10 of 10
25	1000	10 of 10
50	500	10 of 10
50	1000	10 of 10
100	500	10 of 10
100	1000	10 of 10
100	1000	10 of 10

Table 1: Performance of Uncoupled Bilinear Programming Algorithm UBPA on Training Set

Dimension n	Training Set Size	100% Bilinear Separation
5	500	10 of 10
10	500	10 of 10
10	1000	10 of 10
25	500	10 of 10
25	1000	10 of 10

Table 2: Performance of Coupled Bilinear Programming Algorithm BPA on Training Set

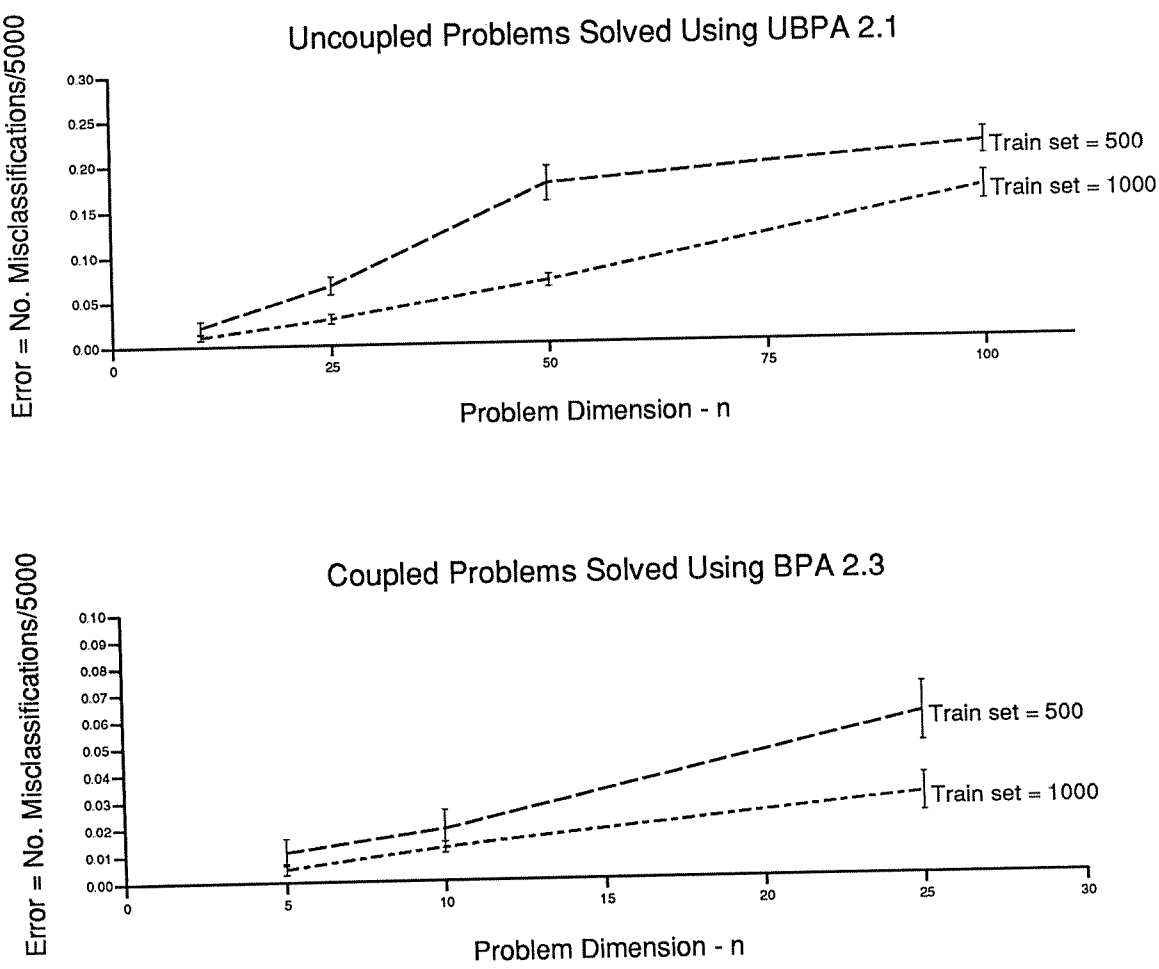


Figure 8: Average Testing Set Error (Testing Set = 5000, 10 trials)

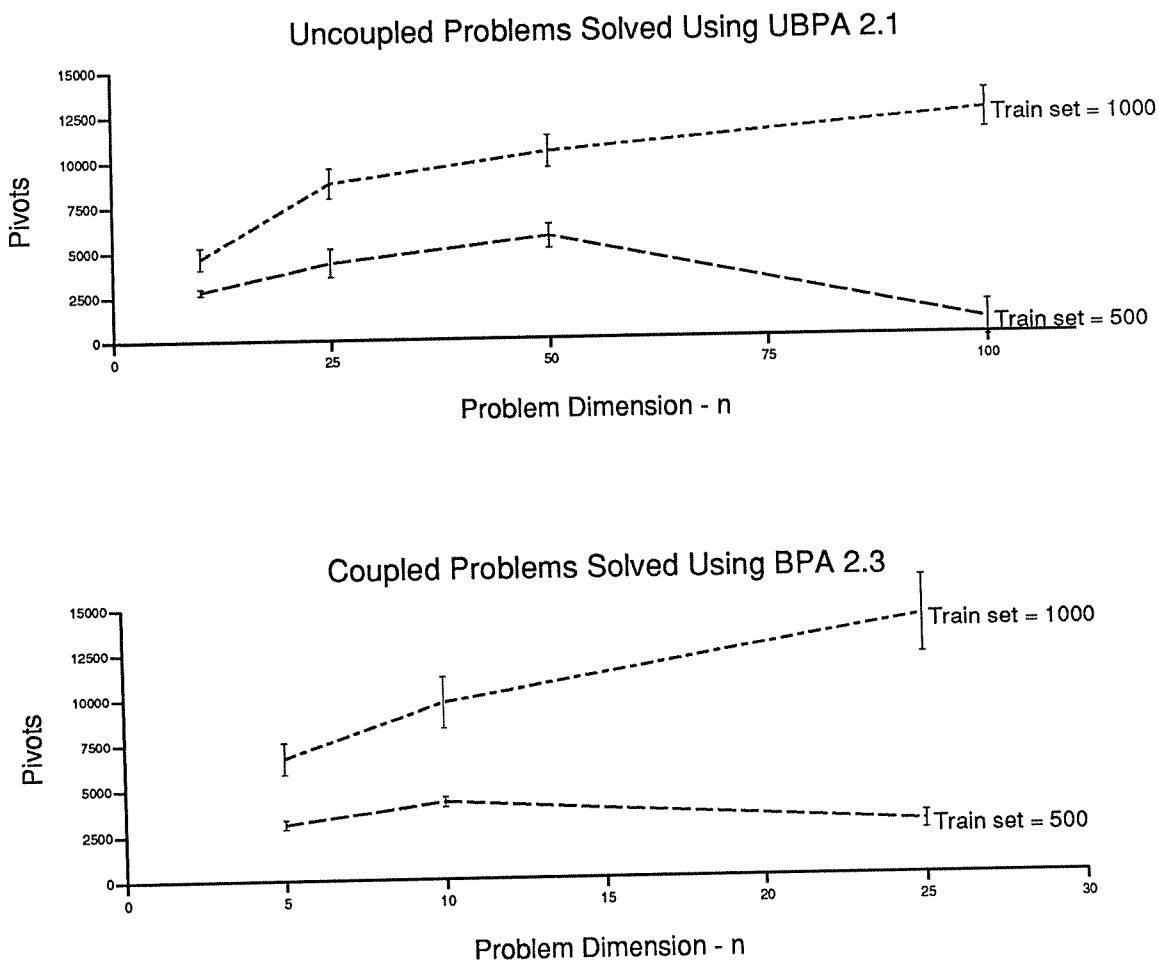


Figure 9: Average Computational Cost (10 trials)

A Frank Wolfe Algorithm for Nonconvex and Bilinear Programs

For convenience and the sake of completeness, we give simple convergence proofs for a Frank-Wolfe [10] algorithm without any convexity assumptions. We also establish finite termination when the sequence of points generated by the algorithm tend to a zero minimum. These results, which do not seem to be readily available in the stated form, are needed for our bilinear program algorithm BPA 2.3. Our proofs are based on those of [7] for the convex case.

We consider the following problem and assumptions.

Problem A.1

$$\min_{x \in \mathcal{X}} f(x)$$

where $f : R^n \to R$, $\mathcal{X}$ is a polyhedral set in R^n that does not contain straight lines that go to infinity in both directions (e.g. it contains the constraint $x \geq 0$), f has continuous first partial derivatives on $\mathcal{X}$ and f is bounded below on $\mathcal{X}$.

Algorithm A.1 (Frank-Wolfe algorithm) *Start with any $x^0 \in \mathcal{X}$. Compute x^{i+1} from x^i as follows.*

- $$v^i \in arg\ vertex \min_{x \in \mathcal{X}} \bigtriangledown f(x^i)x$$
- $$Stop\ if\ \bigtriangledown f(x^i)v^i = \bigtriangledown f(x^i)x^i$$
- $$x^{i+1} = (1 - \lambda^i)x^i + \lambda^i v^i \ where$$
  $$\lambda^i \in arg \min_{0 \leq \lambda \leq 1} f((1 - \lambda)x^i + \lambda v^i)$$

Theorem A.1 (Convergence of Frank-Wolfe algorithm) *The algorithm terminates at some x^j that satisfies the minimum principle necessary optimality condition: $\bigtriangledown f(x^j)(x - x^j) \geq 0$, for all $x \in \mathcal{X}$, or each accumulation point $\bar{x}$ of the sequence $\{x^i\}$ satisfies the minimum principle.*

Proof. If the algorithm stops at x^j then

$$\bigtriangledown f(x^j)x^j = \bigtriangledown f(x^j)v^j = \min_{x \in \mathcal{X}} \bigtriangledown f(x^j)x$$

and the minimum principle is satisfied at x^j. If the algorithm does not terminate, let $\{x^{i_j}\} \to \bar{x}$ and without loss of generality, let $v^{i_j} = \bar{v}$, some fixed vertex of $\mathcal{X}$. Then for $\lambda \in [0, 1]$

$$f((1 - \lambda)x^{i_j} + \lambda\bar{v}) - f(x^{i_j}) \geq \min_{x \in [x^{i_j}, \bar{v}]} f(x) - f(x^{i_j}) = f(x^{i_j+1}) - f(x^{i_j}).$$

Since $\{f(x^i)\}$ is a nonincreasing sequence bounded below it converges. Hence the limit of the last difference above is zero, and we have in the limit

$$f((1 - \lambda)\bar{x} + \lambda\bar{v}) - f(\bar{x}) \geq 0 \quad \forall\ \lambda \in [0, 1]$$

Letting $\lambda \to 0$ and invoking the differentiability of f gives

$$\bigtriangledown f(\bar{x})(\bar{v} - \bar{x}) \geq 0.$$

But by algorithm construction

$$\nabla f(x^{i_j})(x - x^{i_j}) \geq \nabla f(x^{i_j})(\bar{v} - x^{i_j}) \quad \forall \; x \in \mathcal{X}$$

and in the limit

$$\nabla f(\bar{x})(x - \bar{x}) \geq \nabla f(\bar{x})(\bar{v} - \bar{x}) \geq 0 \;\; \forall \, x \in \mathcal{X}$$

□

We establish now finite termination of the Frank-Wolfe algorithm when $f(x)$ is a bilinear nonnegative function with a zero minimum. This is precisely the case for our case of bilinear separability.

Theorem A.2 (Finite termination theorem for Frank-Wolfe algorithm) *In Problem A.1 let*

$$f(x) := (Gx + p)(Hx + q)$$

$$Gx + p \geq 0, \; Hx + q \geq 0 \;\; \forall x \in \mathcal{X}$$

where $G, H \in R^{k \times n}$ and $\mathcal{X}$ is a polyhedral set with no straight lines going to infinity in both directions. If the sequence x^i of the Frank-Wolfe algorithm accumulates to an $\bar{x}$ such that $f(\bar{x}) = 0$, then one of the vertices $\{v^i\}$ of $\mathcal{X}$ generated by the algorithm is a solution. Else $\bar{x}$ satisfies the minimum principle necessary optimality condition.

Proof. Let $\mathcal{V}$ be the finite subset of vertices of $\mathcal{X}$ that constitutes the sequence of vertices $\{v^i\}$ generated by the algorithm. Then

$$\{x^i\} \subset \textit{ convex hull } \{x^0 \cup \mathcal{V}\} \textit{ and } \bar{x} \in \textit{ convex hull } \{x^0 \cup \mathcal{V}\}.$$

where $\bar{x}$ is an accumulation point of $\{x^i\}$ such that $f(\bar{x}) = 0$. If $\bar{x} \in \mathcal{V}$, then we are done. If not then

$$\bar{x} = (1 - \lambda)x + \lambda v, \textit{for some } x \in \mathcal{X}, \; v \in \mathcal{V} \textit{ and } \lambda \in (0, 1)$$

Since for $j = 1, \ldots, k$, one of the linear functions $G_j\bar{x} + p_j$ or $H_j\bar{x} + q_j$ is zero and both are nonnegative at x and v, that same linear function must vanish at both x and v. Hence

$$G_j v + p_j = 0 \textit{ or } H_j v + q_j = 0, \; j = 1, \ldots, k$$

Hence $f(v) = 0$. The last statement of the theorem follows from TheoremA.1. □

References

[1] F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[2] E.B. Baum. Private communication, 1992.

[3] E.B. Baum and D. Haussler. What size net gives valid generalization. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 81–90, San Mateo, California, 1989. Morgan Kaufmann.

[4] K. P. Bennett. Decision tree construction via linear programming. In M. Evans, editor, *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, 1992.

[5] K. P. Bennett and O. L. Mangasarian. Neural network training via linear programming. In P. M. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 56–67, Amsterdam, 1992. North Holland.

[6] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[7] C. Berge and A. Ghouila-Houri. *Programming, Games and Transportation Networks.* Wiley, New York, 1965.

[8] A. Blum and R.L. Rivest. Training a 3-node neural network is np-complete. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 494–501, San Mateo, California, 1989. Morgan Kaufmann.

[9] A. Charnes. Some fundamental theorems of perceptron theory and their geometry. In J. T. Lou and R. H. Wilcox, editors, *Computer and Information Sciences*, pages 67–74, Washington, 1964. Spartan Books.

[10] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[11] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, California, 1991.

[12] W. H. Highleyman. A note on linear separation. *IRE Transactions on Electronic Computers*, 10:777–778, 1961.

[13] O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[14] O.L. Mangasarian. Multi-surface method of pattern separation. *IEEE Transactions on Information Theory*, IT-14:801–807, 1968.

[15] O.L. Mangasarian. *Nonlinear Programming.* McGraw–Hill, New York, 1969.

[16] O.L. Mangasarian. Characterization of linear complementarity problems as linear programs. *Mathematical Programming Study*, 7:74–87, 1978.

[17] O.L. Mangasarian, R. Setiono, and W.H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. In T. F. Coleman and Y. Li, editors, *Proceedings of the Workshop on Large-Scale Numerical Optimization, Cornell University, Ithaca, New York, October 19-20, 1989*, pages 22–31, Philadelphia, Pennsylvania, 1990. SIAM.

[18] J.L. McClelland and D.E. Rummelhart. *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* MIT Press, Cambridge, Massachusetts, 1987.

[19] N. Megiddo. On the complexity of polyhedral separability. *Discrete and Computational Geometry*, 3:325–337, 1988.

[20] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, Massachusetts, 1972.

[21] B.A. Murtagh and M.A. Saunders. MINOS 5.0 user's guide. Technical Report SOL 83.20, Stanford University, December 1983.

[22] K.G. Murty. *Linear Programming.* John Wiley & Sons, New York, New York, 1983.

[23] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[24] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, New York, 1959.

[25] D.E. Rumelhart, G.E. Hinton, and J.L. McClelland. Learning internal representations. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, Massachusetts, 1986. MIT Press.

[26] F. W. Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, C-17:367–372, 1968.

[27] T. V. Thieu. A note on the solution of bilinear programming problems by reduction to concave minimization. *Mathematical Programming*, 41:249–260, 1988.

[28] D. J. White. A linear programming approach to solving bilinear programs. *Mathematical Programming*, 56:45–50, 1992.

[29] W. H. Wolberg and O.L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences, U.S.A.*, 87:9193–9196, 1990.

[30] Y. Yajima and H. Konno. Efficient algorithms for solving rank two and rank three bilinear programming problems. *Journal of Global Optimization*, 1:155–171, 1991.