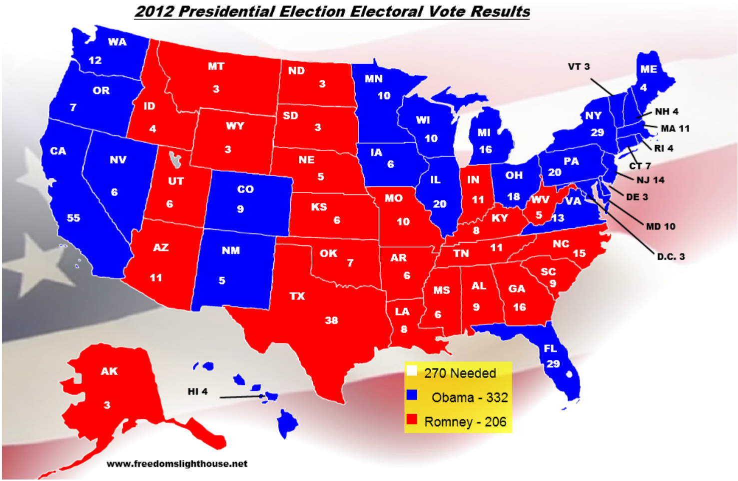
Sigmoidal Programming

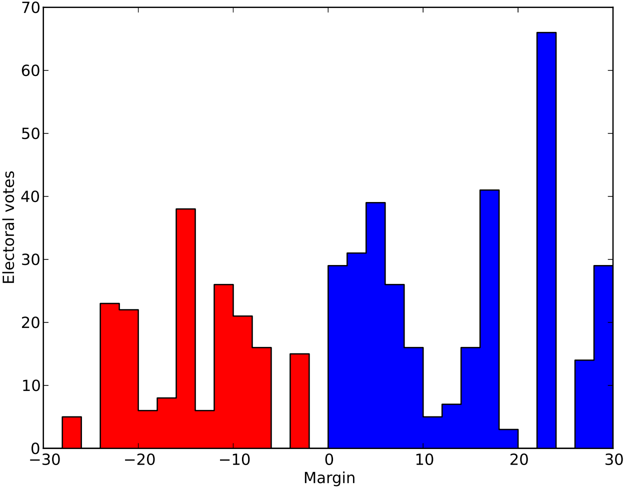
Madeleine Udell and Stephen Boyd
Stanford University
ICME Colloquium

5/13/13

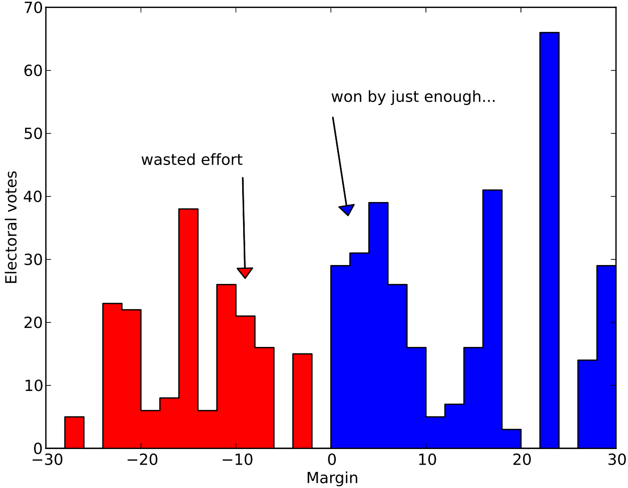
Motivation


2012 Presidential Election Electoral Vote Results
WA 12
OR 7
CA 55
ID 4
NV 6
MT 3
WY 3
UT 6
AZ 11
CO 9
NM 5
ND 3
SD 3
NE 5
KS 6
OK 7
TX 38
MN 10
IA 6
MO 10
AR 6
LA 8
WI 10
IL 20
MI 16
IN 11
OH 18
KY 8
TN 11
MS 6
AL 9
GA 16
FL 29
SC 9
NC 15
VA 13
WV 5
PA 20
NY 29
VT 3
ME 4
NH 4
MA 11
RI 4
CT 7
NJ 14
DE 3
MD 10
D.C. 3
AK 3
HI 4
270 Needed
Obama - 332
Romney - 206
www.freedomslighthouse.net

Optimization on the Obama campaign



Optimization on the Obama campaign



Sigmoidal programming

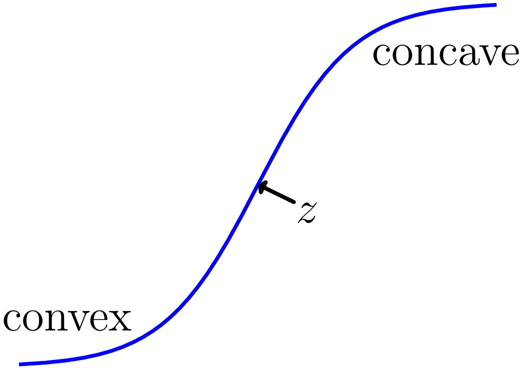
Define the *sigmoidal programming* problem

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

- f_i are *sigmoidal* functions
- $\mathcal{C}$ is a *convex* set of constraints

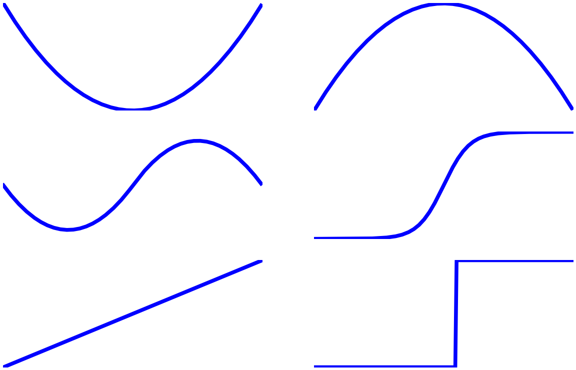
Sigmoidal functions

A continuous function $f : [l, u] \to \mathbf{R}$ is called *sigmoidal* if it is either convex, concave, or convex for $x \le z \in \mathbf{R}$ and concave for $x \ge z$.



Examples of sigmoidal functions

- logistic function $\text{logistic}(x) = 1/(1 + \exp(x))$
- profit function $f(x) = (v - x)\text{logistic}(\alpha x + \beta)$
- admittance function (aproximation to step function)
- CDF of any quasiconcave PDF



Bid optimization

- Each i corresponds to an auction.
- v_i is the value of auction i.
- $p_i(b_i)$ is the probability of winning auction i with bid b_i.
- To maximize winnings (in expectation), choose b by solving

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} v_i p_i(b_i) \\ \text{subject to} & b \in \mathcal{C}. \end{array}$$

- $\mathcal{C}$ represents constraints on our bidding portfolio.

Convex constraints for auctions

- minimum and maximum bids: $l \leq b \leq u$
- budget constraint: $\sum_{i=1}^{n} b_i \leq B$
- sector constraint: $\sum_{i \in S} b_i \leq B_S$
- diversification constraint: $\forall |S| > k$,

$$\sum_{i \in S} b_i \geq \epsilon \sum_{i=1}^{n} b_i$$

- and more (intersections are ok!)

The politician's problem

- Each i corresponds to a *constituency* (*e.g.* state, demographic, ideological group).
- p_i is # votes in constituency i (*e.g.* electoral, popular, etc).
- Politician chooses actions y.
- Constituency i prefers actions w_i.
- $f_i(w_i^T y)$ is the probability of winning constituency i.
- To win the most votes (in expectation), choose y by solving

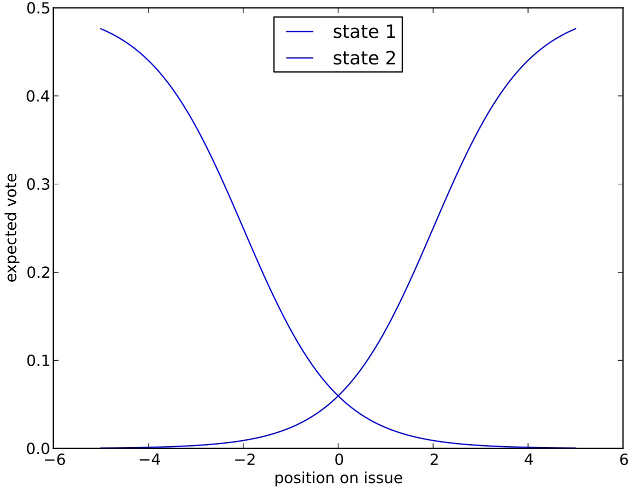
$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n p_i f_i(w_i^T y) \\ \text{subject to} & y \in \mathcal{C}. \end{array}$$

- $\mathcal{C}$ represents constraints on what actions we are willing or able to take.

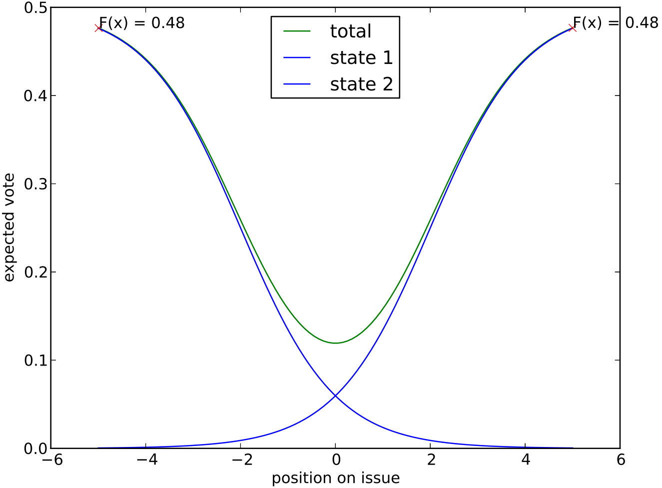
Convex constraints for politicians

- min, max position: $l \leq y \leq u$
- max hrs in day: $\sum_{i=1}^{n} y_i \leq B$
- don't annoy any constituency too much: $w_i^T y \geq -\gamma$

Multiple peaks



Which way to go?



Sigmoidal programming is NP hard

Reduction from integer linear programming:

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

Cast as sigmoidal programming:

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

Optimal value of sigmoidal programming problem is 0 $\iff$ there is an integral solution to $Ax = b$

(Also NP-hard to approximate, using reduction from MAXCUT)

Global optimization

mission: find *all* local maxima

- can't search every *point* in the space
- willing to search a lot of *boxes* — which ones to choose?

Branch and bound

Idea of *branch-and-bound* method (Lawler and Wood, 1968; Balas 1968):

- Partition space into smaller regions $Q \in \mathcal{Q}$.
- Compute upper and lower bounds $U(Q)$ and $L(Q)$ on optimal function value

$$f^\star(Q) = \max_{x \in Q} \sum_{i=1}^{n} f_i(x_i)$$

  in region Q:

$$L(Q) \leq f^\star(Q) \leq U(Q).$$

- Repeat until we zoom in on global max:

$$\max_{Q \in \mathcal{Q}} L(Q) \leq f^\star \leq \max_{Q \in \mathcal{Q}} U(Q).$$

Ingredients for branch and bound success

We need methods to

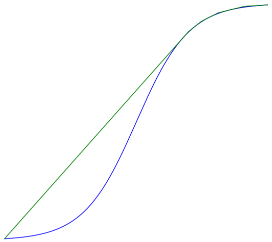
- easily compute upper and lower bounds $U(Q)$ and $L(Q)$,
- choose the next region to split, and
- choose how to split it

so that the bounds become provably tight around the true solution reasonably quickly.

Ingredients for sigmoidal programming

We can do it!

- Easily compute upper and lower bounds $U(Q)$ and $L(Q)$ using *concave envelope* of the functions f_i.
- Choose the region with highest upper bound as the next region to split.
- Split it at the solution to the previous problem along the coordinate with the highest error.



Lower bound

Computing a lower bound is easy, since any feasible point gives a lower bound:

$$\sum_{i=1}^{n} f_i(x_i') \leq \max_{x \in Q} \sum_{i=1}^{n} f_i(x_i) \quad \forall x' \in Q.$$

Upper bound

Upper bound will require solving a maximization problem.

If we have functions $\hat{f}_i$ such that

- $f_i(x_i) \le \hat{f}_i(x_i)$ for every $x \in Q$, and
- $\hat{f}_i$ are all concave.

Then $\sum_{i=1}^n \hat{f}_i(x_i)$ is also concave, so

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

is a convex optimization problem that can be solved efficiently. Let $x^\star$ be a solution to this relaxed problem.

Bound $f^\star(Q)$

Then we have an upper and lower bound on $f^\star(Q)$. Since

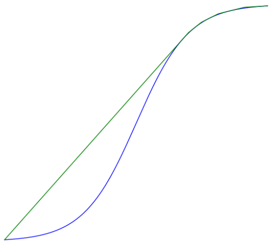
$$\begin{aligned}
f_i(x_i) &\leq \hat{f}_i(x_i) \quad \forall x, i = 1, \ldots, n \\
\sum_{i=1}^{n} f_i(x_i) &\leq \sum_{i=1}^{n} \hat{f}_i(x_i) \quad \forall x \\
\max_{x \in Q} \sum_{i=1}^{n} f_i(x_i) &\leq \max_{x \in Q} \sum_{i=1}^{n} \hat{f}_i(x_i) \\
f^\star(Q) &\leq \max_{x \in Q} \sum_{i=1}^{n} \hat{f}_i(x_i),
\end{aligned}$$

we have that

$$\underbrace{\sum_{i=1}^{n} f_i(x_i^\star)}_{L(Q)} \leq f^\star(Q) \leq \underbrace{\sum_{i=1}^{n} \hat{f}_i(x_i^\star)}_{U(Q)}.$$

Concave envelope

The tightest concave approximation to f is obtained by choosing $\hat{f}_i$ to be the *concave envelope* of the function f.



$\hat{f}_i = -(-f)^{**}$ is the (negative) *bi-conjugate* of $-f$, where

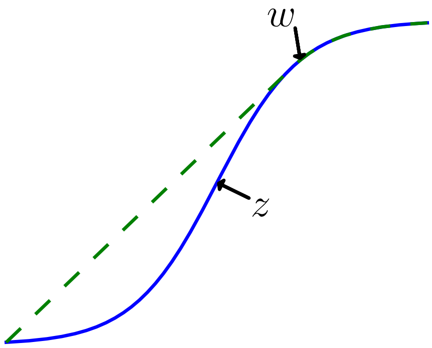
$$f^\star(y) = \sup_{x \in I}(f(x) - yx)$$

is the *conjugate* (also called the *Fenchel dual*) of f.

Concave envelope

The concave envelope can be computed by first locating the point w such that $f(w) = f(a) + f'(w)(w - a)$. Then the concave envelope $\hat{f}$ of f can be written piecewise as

$$\hat{f}(x) = \left\{ \begin{array}{cc} f(a) + f'(w)(x - a) & a \leq x \leq w \\ f(x) & w \leq x \leq b \end{array} \right\}.$$



Branch

Compute tighter approximation by splitting rectangles wisely.

- optimism: split rectangle Q with highest upper bound
- greed: split along coordinate i with greatest error
- hope: split at previous solution $x_i^\star$

Convergence

The number of concave subproblems that must be solved to achieve accuracy ϵ is bounded by

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

where

- $Q_{\text{init}} = (l_1, u_1) \times \cdots \times (l_n, u_n)$,
- $f_i(x) = \int_{l_i}^{x} h_i(t)\, dt,\ i = 1, \ldots, n$, and
- $z_i = \arg\max_{[l_i, u_i]} h_i(x),\ i = 1, \ldots, n$.

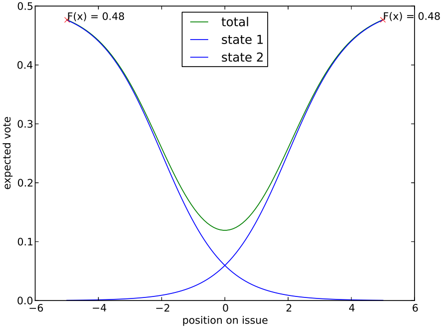
Prune

$$\max_{Q \in \mathcal{Q}} L(Q) \leq f^{\star} \leq \max_{Q \in \mathcal{Q}} U(Q)$$

- Q' is called *active* if $\max_{Q \in \mathcal{Q}} L(Q) \leq U(Q')$.
- Otherwise, the solution cannot lie in Q', and we can ignore it.

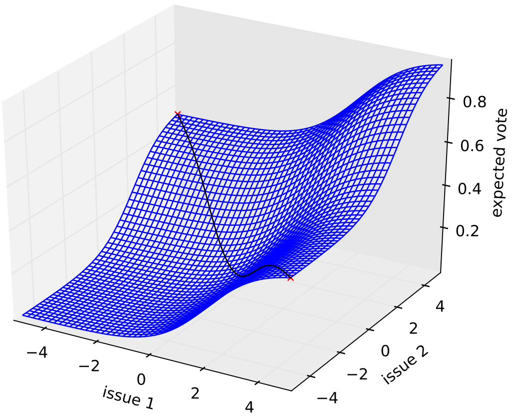
Example: opposing interests

$$\begin{array}{ll} \text{maximize} & \sum_{i=1,2} \text{logistic}(x_i - 2) \\ \text{subject to} & \sum_{i=1,2} x_i = 0 \end{array}$$



Example: opposing interests

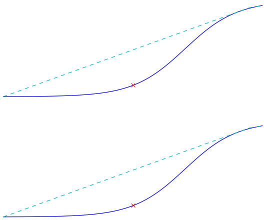
Black line is feasible set.



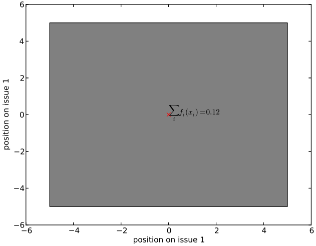
Example: opposing interests

1st iteration:

Best solution so far



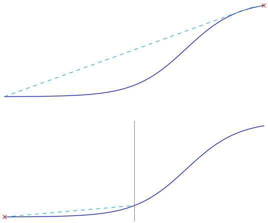
Active nodes



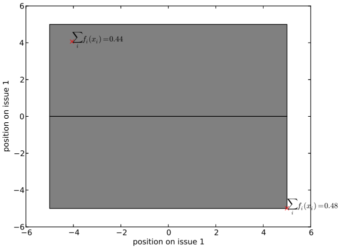
Example: opposing interests

2nd iteration:

Best solution so far



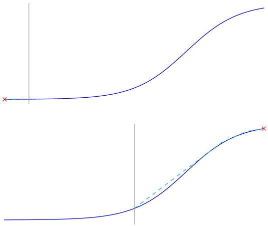
Active nodes

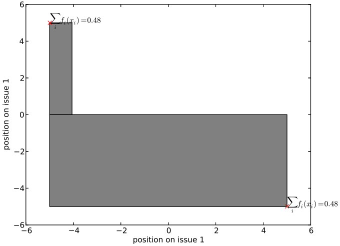


Example: opposing interests

3rd iteration:

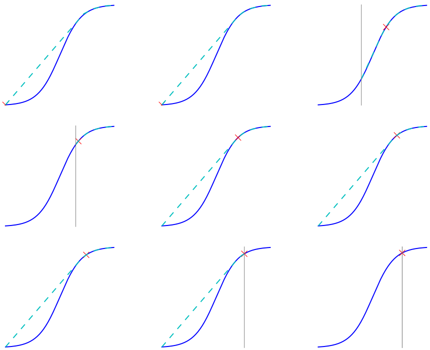
Best solution so far	Active nodes





Bid optimization

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} v_i \text{logistic}(b_i - \beta_i) \\ \text{subject to} & \sum_{i=1}^{n} b_i \leq B \\ & b \geq 0 \end{array}$$



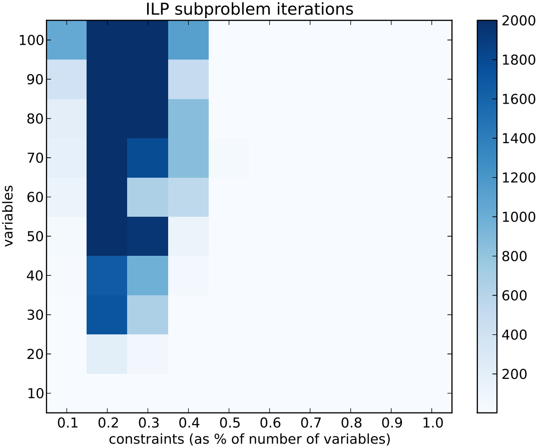
Bid optimization

Simulate for $v_i \sim \mathcal{U}(0,1)$, $B = 1/5 \sum_{i=1}^{n} v_i$, $\beta_i = 1$. To solve to accuracy $.00001 \cdot n$ takes very few steps! (1 step = 1 LP solve)

n	steps
10	2
20	4
30	4
40	6
50	7
60	7
70	6
80	6
90	7
100	6

Integer linear programming

$$
\begin{array}{ll}
\text{maximize} & \sum_{i=1}^{n} |x_i - 1/2| \\
\text{subject to} & Ax = b \\
 & 0 \le x_i \le 1 \quad i = 1, \dots, n
\end{array}
$$



Political positioning: data

- Data comes from 2008 American National Election Survey (ANES).
- Respondents r rate candidates c as having positions $y^{rc} \in [1,7]^m$ on m issues.
- Respondents say how happy they'd be if the candidate won $h^{rc} \in [1,7]$.
- 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 $v^{rc} = 0$.

Political positioning: model

- 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: only 34 states, some with only 14 respondents ...

Political positioning: electoral vote

- Suppose each state i has v_i votes, which they allocate entirely to the winner of the popular vote.
- y denotes the positions the politician takes on the issues.
- Then using our model, the politician's pandering to state i is given by $x_i = w_i^T y$, and the expected number of votes from state i is

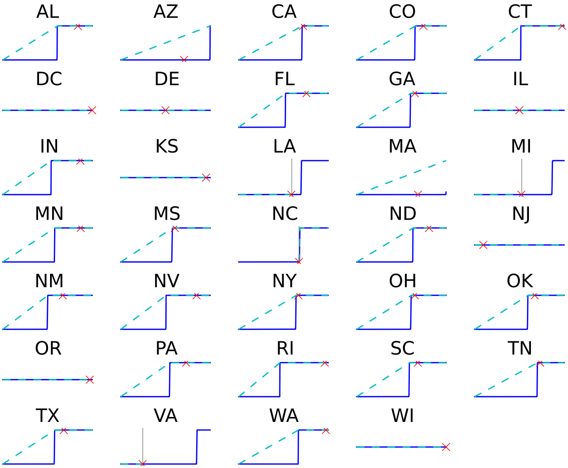
$$v_i \mathbf{1}(\text{logistic}(x_i) > .5),$$

  where $\mathbf{1}(x)$ is 1 if x is true and 0 otherwise.
- Hence the politician will win the most votes if y is chosen by solving

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n v_i \mathbf{1}(\text{logistic}(x_i) > .5) \\ \text{subject to} & x_i = w_i^T y \quad \forall i \\ & 1 \leq y \leq 7. \end{array}$$

Optimal solutions to the politician's problem

Obama's optimal pandering:



Optimal positions for Obama

Issue	Optimal position	Previous position
Spending and 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

Issue	1	7
Spending and services	provide many fewer services	provide many more services
Defense spending	decrease defense spending	increase defense spending
Liberal conservative	liberal	conservative
Assistance to blacks	govt should help blacks	blacks should help themselves

Conclusion

- Sigmoidal programs are a new, non-convex problem class.
- Sigmoidal objectives are ideally suited to model allocation problems with sigmoidal utilities.
- The problem class is NP-hard to approximate to arbitrary precision,
- but solutions on interesting problems are often easy to compute.

Questions?

Network utility maximization

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

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

Network utility maximization

- 500 flows over 500 edges.
- Flows use on average 2.5 edges.
- Each edge has capacity 2.5.
- f_i is an admittance function $i = 1, \ldots, n$.