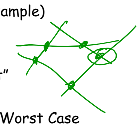
CS 580: Algorithm Design and Analysis

Jeremiah Blocki
Purdue University
Spring 2019

Homework 4: Due tomorrow (March 8) at 11:59 PM

Recap

- Linear Programming
  - Very Powerful Technique (Subject of Entire Courses)
  - **Our Focus:** Using Linear Programming as a Tool
    - Solving Network Flow using Linear Programming
    - Finding Minimax Optimal Strategy in 2-Player Zero Sum Game
    - Operations Research (Brewery Example)
- Solving Linear Programs
  - Simplex Intuition:
    - Optimal point is an "extreme point"
    - No "local optimum"
  - Simplex Runs in Exponential Time in Worst Case
    - But other algorithms (e.g., Ellipsoid) run in polynomial time



2

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

Shooter scores 80% of time when shooter aims right and goalie blocks left

Minimax Optimal Strategy (possibly randomized) best strategy you can find given that opponent is rational (and knows your strategy)

How can we find Minimax Optimal Strategy?

3

2-Player Zero-Sum Games

Player 1 Mixed Strategy: $p_1, \ldots, p_n$ **(n actions)**

Suppose Player 2 plays action j

→Player 1 receives reward m_{ij} with probability p_i
→ Player 1 receives expected reward $\sum_i p_i m_{ij}$ when player 2 plays j

→ Player 2 receives expected reward $-\sum_i p_i m_{ij}$

Player 2 will select j to maximize $-\sum_i p_i m_{ij}$
(Equivalently, minimize $\sum_i p_i m_{ij}$)

Therefore, expected value of Strategy $p_1, \ldots, p_n$ to player 1 is

$$V_1(p_1, \ldots, p_n) = \min_j \left\{ \sum_i p_i m_{ij} \right\}$$

Best Strategy $p_1^*, \ldots, p_n^*$ should maximize $V_1^* = V_1(p_1^*, \ldots, p_n^*)$

4

Finding Minimax Optimal Solution using Linear Programming

Variables: $p_1, \ldots p_n$ and v (p_i is probability of action i)
Goal: Maximize v (our expected reward).

Constraints:
- $p_1, \ldots, p_n \geq 0$
- $1 \geq p_1 + \ldots + p_n \geq 1$
- For all columns j we have

$$\sum_i p_i m_{ij} \geq v$$

Expected reward when player 2 takes action j

m_{ij} denotes reward when player 1 takes action i and player 2 takes action j.

Duality: $V_1^* = -V_2^*$

5

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

Shooter Minimax (solve in Mathematica)

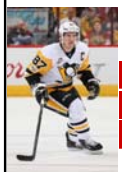
Pr shoot left/right

$$\text{Maximize}\left[\left\{ \begin{matrix} v, \\ S_L \geq 0, S_R \geq 0, 1 \leq S_L + S_R \leq 1, \\ \underbrace{S_L\left(\frac{1}{2}\right) + S_R\left(\frac{4}{5}\right) \geq v}_{\text{Goalie Blocks Left}}, \quad \underbrace{S_L\left(\frac{9}{10}\right) + S_R\left(\frac{1}{3}\right) \geq v}_{\text{Goalie Blocks Right}} \end{matrix} \right\}, \{v, S_L, S_R\}\right]$$

6

2-Player Zero-Sum Games

Example: Shooter-Goalie



	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ S_L \geq 0, S_R \geq 0, 1 \leq S_L + S_R \leq 1, \\ S_L(1/2) + S_R(4/5) \geq v, \\ S_L(9/10) + S_R(1/3) \geq v \end{array}\right\}, \{v, S_L, S_R\}\right]$$

Solution: {v->0.638462,L->0.538462,R->0.461538}

- Shooter guaranteed to score *at least* 63.846% of the time by shooting left 53.846% of time

7

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

Shooter Minimax (solve in Mathematica)

Pr block left/right

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ B_L \geq 0, B_R \geq 0, \quad 1 \leq B_L + B_R \leq 1, \\ \underbrace{B_L\left(-\frac{1}{2}\right) + B_R\left(-\frac{9}{10}\right) \geq v}_{\text{Shooter Shoots Left}}, \quad \underbrace{B_L\left(-\frac{4}{5}\right) + B_R\left(-\frac{1}{3}\right) \geq v}_{\text{Shooter Shoots Right}} \end{array}\right\}, \{v, B_L, B_R\}\right]$$

8

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ B_L \geq 0, B_R \geq 0, 1 \leq B_L + B_R \leq 1, \\ B_L(-1/2) + B_R(-9/10) \geq v, \\ B_L(-4/5) + B_R(-1/3) \geq v \end{array}\right\}, \{v, B_L, B_R\}\right]$$

Solution: {v->-0.638462,B_L->0.653846,B_R->0.346154}

- Goal scored *at most* 63.846% of time if goalie blocks right 65.384% of the time

9

Chapter 8

NP and Computational Intractability

Algorithm Design Patterns and Anti-Patterns

Algorithm design patterns.	Ex.
▪ Greedy.	O(n log n) interval scheduling.
▪ Divide-and-conquer.	O(n log n) FFT.
▪ Dynamic programming.	O(n^2) edit distance.
▪ Duality.	O(n^3) bipartite matching.
▪ Reductions.	Circulation via Network Flow
	Bipartite Matching via Network Flow
	Minimax Strategy via Linear Programming
▪ Local search.	
▪ Randomization.	

Algorithm design anti-patterns.	
▪ NP-completeness.	O(n^k) algorithm unlikely.
▪ PSPACE-completeness.	O(n^k) certification algorithm unlikely.
▪ Undecidability.	No algorithm possible.

11

8.1 Polynomial-Time Reductions

Classify Problems According to Computational Requirements

Q. Which problems will we be able to solve in practice?

A working definition. [von Neumann 1953, Godel 1956, Cobham 1964, Edmonds 1965, Rabin 1966]
Those with polynomial-time algorithms.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality testing	Factoring

13

Classify Problems

Desiderata. Classify problems according to those that can be solved in polynomial-time and those that cannot.

Provably requires exponential-time.
- Given a Turing machine, does it halt in at most k steps?
- Given a board position in an n-by-n generalization of chess, can black guarantee a win?

Frustrating news. Huge number of fundamental problems have defied classification for decades.

This chapter. Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one really hard problem.

14

Polynomial-Time Reduction

Desiderata'. Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?

don't confuse with reduces from

Reduction. Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

Notation. $X \leq_P Y$.

computational model supplemented by special piece of hardware that solves instances of Y in a single step

Remarks.
- We pay for time to write down instances sent to black box $\Rightarrow$ instances of Y must be of polynomial size.
- Note: Cook reducibility.

in contrast to Karp reductions

15

Polynomial-Time Reduction

Purpose. Classify problems according to relative difficulty.

Design algorithms. If $X \leq_P Y$ and Y can be solved in polynomial-time, then X can also be solved in polynomial time.

Establish intractability. If $X \leq_P Y$ and X cannot be solved in polynomial-time, then Y cannot be solved in polynomial time.

Establish equivalence. If $X \leq_P Y$ and $Y \leq_P X$, we use notation $X \equiv_P Y$.

up to cost of reduction

16

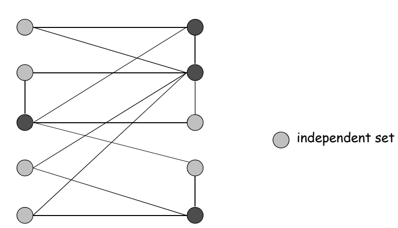
Reduction By Simple Equivalence

Basic reduction strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Independent Set

INDEPENDENT SET: Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S?

Ex. Is there an independent set of size ≥ 6? Yes.
Ex. Is there an independent set of size ≥ 7? No.

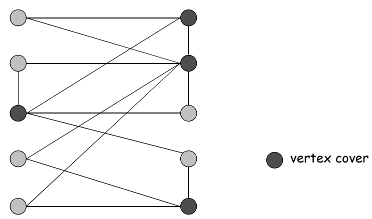


18

Vertex Cover

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?

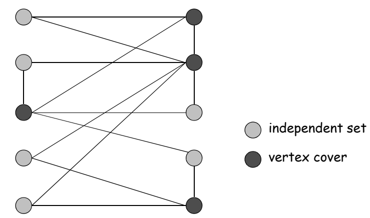
Ex. Is there a vertex cover of size ≤ 4? Yes.
Ex. Is there a vertex cover of size ≤ 3? No.



19

Vertex Cover and Independent Set

Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
Pf. We show S is an independent set iff V – S is a vertex cover.



20

Vertex Cover and Independent Set

Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
Pf. We show S is an independent set iff V – S is a vertex cover.

$\Rightarrow$
- Let S be any independent set.
- Consider an arbitrary edge (u, v).
- S independent $\Rightarrow u \notin S$ or $v \notin S$ $\Rightarrow$ $u \in V - S$ or $v \in V - S$.
- Thus, V – S covers (u, v).

$\Leftarrow$
- Let V – S be any vertex cover.
- Consider two nodes $u \in S$ and $v \in S$.
- Observe that $(u, v) \notin E$ since V – S is a vertex cover.
- Thus, no two nodes in S are joined by an edge $\Rightarrow$ S independent set. ▪

21

Reduction from Special Case to General Case

Basic reduction strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover

SET COVER: Given a set U of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and an integer k, does there exist a collection of $\leq k$ of these sets whose union is equal to U?

Sample application.
- m available pieces of software.
- Set U of n capabilities that we would like our system to have.
- The ith piece of software provides the set $S_i \subseteq U$ of capabilities.
- Goal: achieve all n capabilities using fewest pieces of software.

Ex:

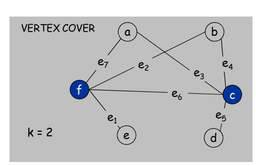
U = { 1, 2, 3, 4, 5, 6, 7 }	
k = 2	
S_1 = {3, 7}	S_4 = {2, 4}
S_2 = {3, 4, 5, 6}	S_5 = {5}
S_3 = {1}	S_6 = {1, 2, 6, 7}

23

Vertex Cover Reduces to Set Cover

Claim. VERTEX-COVER $\leq_P$ SET-COVER.
Pf. Given a VERTEX-COVER instance G = (V, E), k, we construct a set cover instance whose size equals the size of the vertex cover instance.

Construction.
- Create SET-COVER instance:
  - $k = k$, $U = E$, $S_v = \{e \in E : e \text{ incident to } v\}$
- Set-cover of size $\leq k$ iff vertex cover of size $\leq k$. ▪



SET COVER

U = { 1, 2, 3, 4, 5, 6, 7 }	
k = 2	
S_a = {3, 7}	S_b = {2, 4}
S_c = {3, 4, 5, 6}	S_d = {5}
S_e = {1}	S_f = {1, 2, 6, 7}

24

Polynomial-Time Reduction

Basic strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

8.2 Reductions via "Gadgets"

Basic reduction strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction via "gadgets."

Satisfiability

Literal: A Boolean variable or its negation. x_i or $\overline{x_i}$

Clause: A disjunction of literals. $C_j = x_1 \vee \overline{x_2} \vee x_3$

Conjunctive normal form: A propositional formula Φ that is the conjunction of clauses. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

SAT: Given CNF formula Φ, does it have a satisfying truth assignment?

3-SAT: SAT where each clause contains (*at most*) 3 literals.

each corresponds to a different variable

Ex: $\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_2 \vee x_3\right) \wedge \left(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}\right)$

Yes: x_1 = true, x_2 = true x_3 = false.

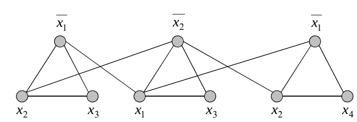
3 Satisfiability Reduces to Independent Set

Claim. 3-SAT $\le_P$ INDEPENDENT-SET.

Pf. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

Construction.
- G contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

G



k = 3 $\quad \Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$

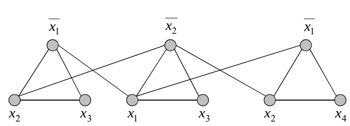
3 Satisfiability Reduces to Independent Set

Claim. G contains independent set of size k = $|\Phi|$ iff Φ is satisfiable.

Pf. $\Rightarrow$ Let S be independent set of size k.
- S must contain exactly one vertex in each triangle.
- Set these literals to true. ← and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

Pf $\Leftarrow$ Given satisfying assignment, select one true literal from each triangle. This is an independent set of size k. ▪

G



k = 3 $\quad \Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$

Review

Basic reduction strategies.
- Simple equivalence: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
- Special case to general case: VERTEX-COVER $\le_P$ SET-COVER.
- Encoding with gadgets: 3-SAT $\le_P$ INDEPENDENT-SET.

Transitivity. If X $\le_P$ Y and Y $\le_P$ Z, then X $\le_P$ Z.
Pf idea. Compose the two algorithms.

Ex: 3-SAT $\le_P$ INDEPENDENT-SET $\le_P$ VERTEX-COVER $\le_P$ SET-COVER.

Self-Reducibility

Decision problem. Does there exist a vertex cover of size $\leq$ k?
Search problem. Find vertex cover of minimum cardinality.

Self-reducibility. Search problem $\leq_P$ decision version.
- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.
- (Binary) search for cardinality k* of min vertex cover.
- Find a vertex v such that $G - \{v\}$ has a vertex cover of size $\leq$ k* - 1.
  - any vertex in any min vertex cover will have this property
- Include v in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$. delete v and all incident edges

31

8.3 Definition of NP

Decision Problems

Decision problem.
- X is a set of strings.
- Instance: string s.
- Algorithm A solves problem X: A(s) = yes iff $s \in X$.

Polynomial time. Algorithm A runs in poly-time if for every string s, A(s) terminates in at most $p(|s|)$ "steps", where $p(\cdot)$ is some polynomial.
length of s

PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }
Algorithm. [Agrawal-Kayal-Saxena, 2002] $p(|s|) = |s|^8$.

33

Definition of P

P. Decision problems for which there is a poly-time algorithm.

Problem	Description	Algorithm	Yes	No
MULTIPLE	Is x a multiple of y?	Grade school division	51, 17	51, 16
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5?	Dynamic programming	niether neither	acgggt ttttta
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

34

NP

Certification algorithm intuition.
- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof t that $s \in X$.

Def. Algorithm C(s, t) is a certifier for problem X if for every string s, $s \in X$ iff there exists a string t such that C(s, t) = yes.
"certificate" or "witness"

NP. Decision problems for which there exists a poly-time certifier.
C(s, t) is a poly-time algorithm and $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

Remark. NP stands for nondeterministic polynomial-time.

35

Certifiers and Certificates: Composite

COMPOSITES. Given an integer s, is s composite?

Certificate. A nontrivial factor t of s. Note that such a certificate exists iff s is composite. Moreover $|t| \leq |s|$.

Certifier.
```
boolean C(s, t) {
   if (t ≤ 1 or t ≥ s)
      return false
   else if (s is a multiple of t)
      return true
   else
      return false
}
```

Instance. s = 437,669.
Certificate. t = 541 or 809. ← 437,669 = 541 × 809

Conclusion. COMPOSITES is in NP.

36

Certifiers and Certificates: 3-Satisfiability

SAT. Given a CNF formula Φ, is there a satisfying assignment?

Certificate. An assignment of truth values to the n boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

Ex.

$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

instance s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

certificate t

Conclusion. SAT is in NP.

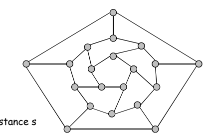
Certifiers and Certificates: Hamiltonian Cycle

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?

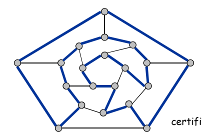
Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Conclusion. HAM-CYCLE is in NP.



instance s



certificate t

P, NP, EXP

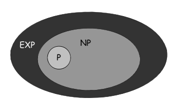
P. Decision problems for which there is a poly-time algorithm.
EXP. Decision problems for which there is an exponential-time algorithm.
NP. Decision problems for which there is a poly-time certifier.

Claim. P $\subseteq$ NP.
Pf. Consider any problem X in P.
- By definition, there exists a poly-time algorithm A(s) that solves X.
- Certificate: t = ε, certifier C(s, t) = A(s). ▪

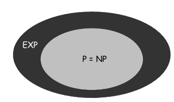
Claim. NP $\subseteq$ EXP.
Pf. Consider any problem X in NP.
- By definition, there exists a poly-time certifier C(s, t) for X.
- To solve input s, run C(s, t) on all strings t with $|t| \le p(|s|)$.
- Return yes, if C(s, t) returns yes for any of these. ▪

The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.



If P ≠ NP



If P = NP

would break RSA cryptography
(and potentially collapse economy)

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …
If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …

Consensus opinion on P = NP? Probably no.

The Simpson's: P = NP?

Futurama: P = NP?

Looking for a Job?

Some writers for the Simpsons and Futurama.

- J. Steward Burns. M.S. in mathematics, Berkeley, 1993.
- David X. Cohen. M.S. in computer science, Berkeley, 1992.
- Al Jean. B.S. in mathematics, Harvard, 1981.
- Ken Keeler. Ph.D. in applied mathematics, Harvard, 1990.
- Jeff Westbrook. Ph.D. in computer science, Princeton, 1989.

43

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
Shoot Right	0.8	1/3

Shooter Minimax (solve in Mathematica)

Pr shoot left/right

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ L_s \geq 0 \,\&\&\, R_s \geq 0 \,\&\&\, L_s + R_s == 1 \,\&\& \\ \underbrace{L_s\left(\frac{1}{2}\right) + R_s\left(\frac{4}{5}\right)}_{\text{Goalie Blocks Left}} \geq v \,\&\&\, \underbrace{L_s\left(\frac{9}{10}\right) + R_s\left(\frac{1}{3}\right)}_{\text{Goalie Blocks Right}} \geq v \end{array}\right\}, \{v, L_s\right.$$

44

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ L \geq 0, R \geq 0, 1 \leq L + R \leq 1, \\ L(1/2) + R(4/5) \geq v, \\ L(9/10) + R(1/3) \geq v \end{array}\right\}, \{v, L, R\}\right]$$

Solution:
{v->0.638462,L->0.538462,R->0.461538}

Shooter guaranteed to score *at least* 63.846% of the time by shooting left 53.846% of time

45

2-Player Zero-Sum Games

Example: Shooter-Goalie

	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

Shooter Minimax (solve in Mathematica)

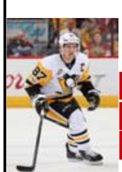
Pr block left/right

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ L_G \geq 0 \,\&\&\, R_G \geq 0 \,\&\&\, L_G + R_G == 1 \,\&\& \\ \underbrace{L_G\left(\frac{-1}{2}\right) + R_G\left(\frac{-9}{10}\right)}_{\text{Shooter Shoots Left}} \geq v \,\&\&\, \underbrace{L_G\left(\frac{-4}{5}\right) + R_G\left(\frac{-1}{3}\right)}_{\text{Shooter Shoots Right}} \geq v \end{array}\right\}, \{v, L_G, R_G\}\right]$$

46

2-Player Zero-Sum Games

Example: Shooter-Goalie



	Block Left	Block Right
Shoot Left	1/2	0.9
shoot Right	0.8	1/3

$$\text{Maximize}\left[\left\{\begin{array}{c} v, \\ L_G \geq 0 \,\&\&\, R_G \geq 0 \,\&\&\, L_G + R_G == 1 \,\&\& \\ L_G\left(\frac{-1}{2}\right) + R_G\left(\frac{-9}{10}\right) \geq v \,\&\&\, L_G\left(\frac{-4}{5}\right) + R_G\left(\frac{-1}{3}\right) \geq v \end{array}\right\}, \{v, L_G, R_G\}\right]$$

Solution:
$\{v \to -0.638462, L_G \to 0.653846, R_G \to 0.346154\}$

Goalie can ensure goal is scored *at most* 63.846% of the time by blocking left 65.384% of the time

47