Linear Programming for Approximation Algorithms

A quick overview of basics needed to understand and apply linear programming in approximation algorithms

A functional approach and biased towards particular needs of class

Linear Programming

An optimization problem on n real valued variables $x_1, x_2, \ldots, x_n$

Objective function and constraints are linear in the variables

Standard form

min $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$

subject to:

$a_{11} x_1 + \ldots + a_{1n} x_n \geq b_1$

$a_{21} x_1 + \ldots + a_{2n} x_n \geq b_2$

...

$a_{m1} x_1 + \ldots + a_{mn} x_n \geq b_m$

$x_1, x_2, \ldots, x_n \geq 0$

Standard form

In matrix form:

min $c\,x$

subject to

$A\,x \geq b$

$x \geq 0$

n : dimension of the problem

m : number of rows/constraints in A

All LP problems can be reduced to the standard form in polynomial time (and polynomial blow up in m, n)

Solving LP Problems

LP Problem satisfies exactly one of the following properties:

- has a finite optimum
- is infeasible
- is unbounded

Given *rational valued* c, A, b there is a *polynomial* time algorithm to solve the problem; that is decide the right property above and output the finite optimum if it has one

Solving LP Problems

Given *rational valued* c, A, b there is a *polynomial* time algorithm to solve the problem

In particular the above implies that there exists an optimum solution x^* whose representation is of *size polynomial in the input size*

Basic solutions

The feasible region (all $x \in \mathcal{R}^n$ that satisfy the constraints $Ax \leq b$, $x \geq 0$) is *convex:*

x, y feasible implies $\lambda x + (1-\lambda)y$ also feasible for all $\lambda \in [0,1]$

The feasible region is called a polyhedron

If it is bounded then it is a polytope

The polyhedron need not be a polytope for the problem to have a finite optimum. It depends on c

For every unbounded polyhedron there is a c s.t the optimum is not finite

Basic solutions

A vertex of the polyhedron is a point that is the intersection of n inequalities (halfspaces)

Vertices are also called *basic solutions*

If the LP problem has a finite optimum then it has an optimum solution x^* where x^* is a vertex of the polyhedron

(vertex solutions have important properties that can be exploited in algorithms)

Duality

Every LP problem has a *dual* LP problem

The dual of the dual is the original LP (the primal)

The two LP problems are often referred to as the primal-dual pair

In the standard form:

primal:

min $c\,x$

$A\,x \geq b$

$x \geq 0$

dual:

max $y\,b$

$y\,A \leq c$

$y \geq 0$

Duality

primal: n variables, m constraints (rows of A)

dual: m variables (one for each constraint in primal), n contraints (one for each variable in primal)

Weak duality: if x' is a feasible soln to primal and y' is a feasible soln to dual then

$$c\, x' \geq y'\, b$$

Duality

Weak duality: if x' is a feasible soln to primal and y' is a feasible soln to dual then
$c\,x' \geq y'\,b$

since $y'\,A \leq c$ and $x' \geq 0$, $y'\,A\,x' \leq c\,x'$
but $A\,x' \geq b$ and $y' \geq 0$ therefore $y'\,A\,x' \geq y'\,b$

Corollary: one of the following holds

- both primal and dual have finite optimum
- primal is unbounded and dual is infeasible
- primal is infeasible and dual is unbounded

Strong Duality

If x^* and y^* are finite optima for primal and dual then $c\, x^* = y^*\, b$

Moreover

Complementary slackness:

(primal complementary slackness)
for $1 \leq i \leq n$, if $x^*_i > 0$ then $y^* A_i = c_i$

(dual complementary slackness)
for $1 \leq j \leq m$, if $y^*_j > 0$ then $A_j\, x^* = b_j$

Algorithms to solve LPs

Typical LP problem:

c, A, b given explicitly

- Simplex algorithm(s): practical, widely used, can take exponential time in worst case
- Ellipsoid algorithm(s): impractical, not used, very useful in theory, polynomial time algorithm
- Interior point algorithm(s): practical, used for some large problems, polynomial time algorithm

LP in Approximation Algorithms

Integer (linear) programming (IP) problem: same as LP but the variables $x_1, \ldots, x_n$ are constrained to be *integers*

IP is *NP-Hard* (Why?)

Feasibility question of IP is in *NP* (this fact needs some non-trivial work. Why?)

IP can be solved in poly-time for fixed # of variables (# of constraints need not be fixed). This is a non-trivial result.

LP in Approximation Algorithms

In applications to approximation algorithms and combinatorial optimization we are mostly interested in 0,1 IP where the variables are constrained to be integers in {0, 1}

0,1 IP is NP-Hard. Trivial to see that feasibility problem of 0,1 IP is in NP.

LP Relaxations

Since IP is NP-hard *any* NPO problem can be written (or reduced to) as an IP problem

We obtain a LP *relaxation* by letting the variables in the IP problem take on real values

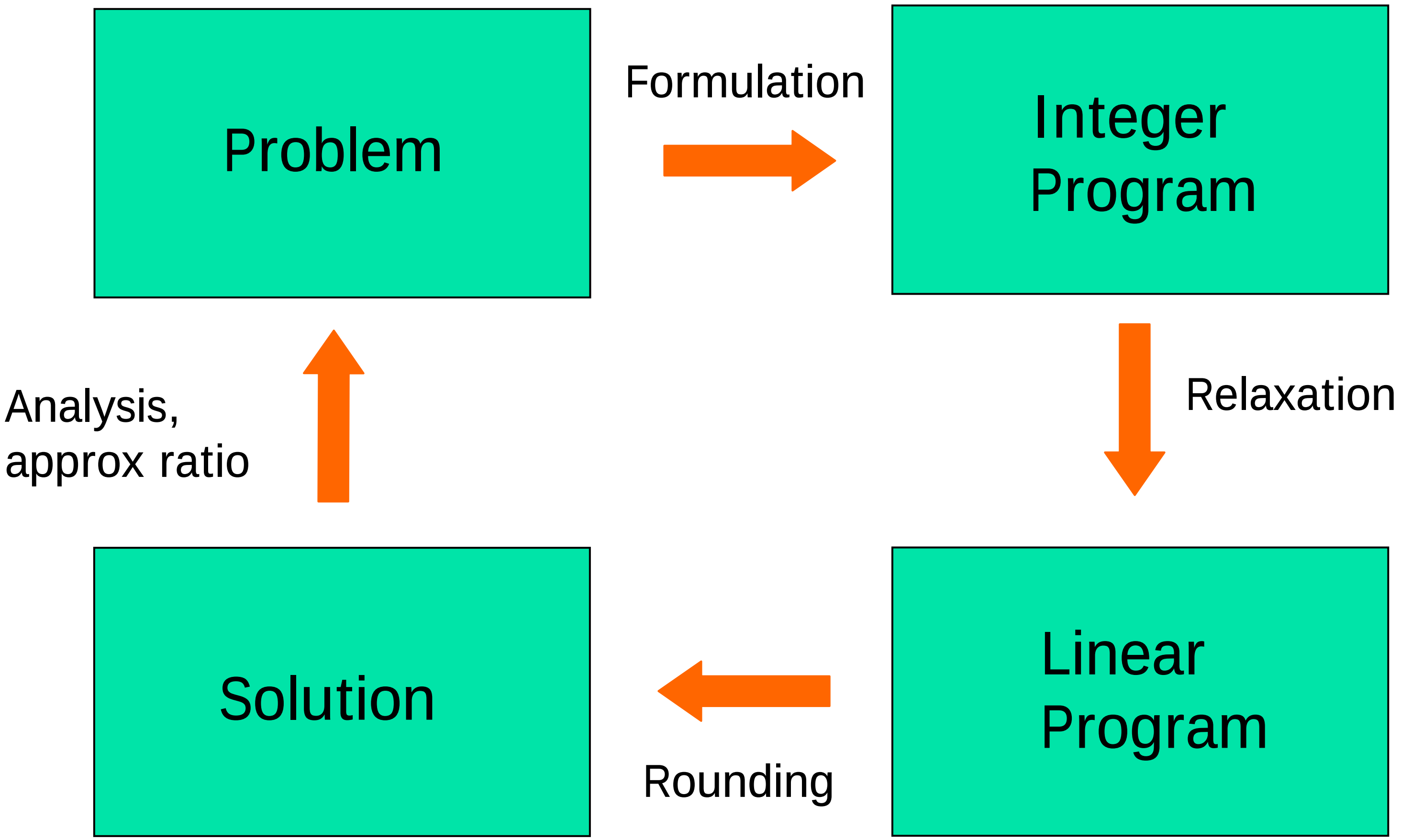
More precisely: Let Π be an NPO problem. Then there is a poly-time computable reduction f such that each instance I of Π can be reduced to an IP problem f(I). We obtain an LP problem f'(I) by letting the variables be real valued

LP Relaxations

More precisely: Let Π be an NPO problem. Then there is a poly-time computable reduction f such that each instance I of Π can be reduced to an IP problem f(I). We obtain an LP problem f'(I) by letting the variables be real valued

The function f is usually called a *formulation* and is typically guided by Π

LP in Approximation Algorithms



LP in Approximation Algorithms

An IP formulation naturally leads from an instance I of Π to an LP problem f'(I). Note that the size of f'(I) is polynomial in the size of I. We can be solve the LP problem f'(I) in polynomial time

OPT(I) : value of an optimum solution to I

$OPT_{LP}(I)$: value of an optimum solution to f'(I)

for minimization problems $OPT_{LP}(I) \leq OPT(I)$

for maximization $OPT_{LP}(I) \geq OPT(I)$

Integrality gap

For a formulation integrality gap is

$\sup_I OPT(I)/OPT_{LP}(I)$

that is, the worst case gap between the (integer) optimum and the fractional optimum

Approximation via LP

- Find good formulations
- Prove constructive (algorithmic) bounds on integrality gap
- Translate into effective algorithms

Pros of LP approach to approximation

- Generic paradigm that applies to all NPO problems
- Solution to LP gives both a lower bound ($OPT_{LP}(I)$) on OPT(I) (in case of minimization) as well as useful information to convert fractional solution (round) into an integer solution.
- For many problems solution quality much better than guaranteed by integrality gaps
- Often LP can be solved faster for the problem at hand or insight leads to a combinatorial algorithm that is much faster in practice.

Cons of LP approach to approximation

- LPs are not easy to solve quickly although polynomial time algorithms exist. Numerical issues (not strongly polynomial time). Typical formulations have large size. Infeasible in some cases.

- Does not completely eliminate the search for a good formulation (algorithm).

Art/techniques for rounding

Typically one proves an upper bound on integrality gap of a formulation by exhibiting an algorithm that rounds a fractional solution to the LP in to an integer solution. The analysis of the rounding gives a bound on the integrality gap.

How does one round?

Art/techniques for rounding

Some general techniques will be explored in the class via various problems

- Primal approach: use a solution the LP and transform it directly into an integer solution
  - randomized rounding
  - iterative rounding
  - decomposition
- Dual approach: use the dual of LP in some way
  - dual-fitting.
  - primal-dual

Vertex Cover via LP

Vertex Cover: given $G=(V,E)$ and $w: V \to \mathcal{R}^+$, find a min weight set of vertices that cover (incident to) all edges

IP formulation: binary variable x(i) for each vertex i in V. x(i) = 1 to indicate that i is chosen in cover, x(i) = 0 to indicate that i is not chosen

min $\sum_{i \in V} w(i)\, x(i)$

s.t $\quad x(i) + x(j) \geq 1$ for each edge $ij \in E$

$\quad x(i) \in \{0, 1\}$ for each $i \in V$

Vertex Cover via LP

IP

min $\sum_{i \in V} w(i)\, x(i)$

s.t $\quad x(i) + x(j) \geq 1$ for each edge $ij \in E$

$\quad x(i) \in \{0, 1\}$ for each $i \in V$

LP relaxation:

min $\sum_{i \in V} w(i)\, x(i)$

s.t $\quad x(i) + x(j) \geq 1$ for each edge $ij \in E$

$\quad x(i) \in [0, 1]$ for each $i \in V$

Rounding the LP

Solve LP to obtain an optimum (fractional) solution x^*
Let $S = \{ i \mid x^*(i) \geq ½ \}$

Output S

Claim: S is a feasible vertex cover

consider any edge ij. By feasibility of x^*,
$x^*(i) + x^*(j) \geq 1$ and hence either $x^*(i) \geq ½$ or $x^*(j) \geq ½$.

One of i, j will be in S

Rounding the LP

Claim: $w(S) \leq 2\ OPT_{LP}$

$w(S) = \sum_{i \in S} w(i)$

$\leq 2 \sum_{i \in S} w(i)\ x^*(i)$ (since $x^*(i) \geq ½$ for $i \in S$)

$\leq 2 \sum_{i \in V} w(i)\ x^*(i)$

$= 2\ OPT_{LP}$

Therefore $OPT_{LP}(I) \geq OPT(I)/2$ for all I, hence integrality gap of formulation is at most 2

Note: rounding works with any feasible solution, hence an approximate optimum soln for LP is sufficient

VC in bipartite graphs

By the famour Konig's theorem, in bipartite graphs the size of a minimum vertex cover (unweighted) is equal to the size of a maximum matching. Therefore one can compute the optimum value for unweighted VC in bipartite graphs.

Moreover by total unimodularity of the edge-vertex incidence matrix in bipartite graphs, the LP for VC has integer solutions so the weighted case can also be solved optimally.

Example for integrality gap of 2

Need to use non-bipartite graphs

Simple example. G is triangle (complete graph on 3 vertices: K_3)

$OPT = 2$

$OPT_{LP} = 3/2$

For K_n, $OPT = n-1$, $OPT_{LP} = n/2$ so gap is $2-1/n$ which tends to 2 as n tends to infinity

Vertex Cover

Current best approximation ratio for VC is

$2 - \Theta(1/\sqrt{\log n})$ $(2 - o(1))$

Outstanding open problem: obtain a $2-\varepsilon$ approximation *or* to prove that it is NP-hard to obtain $2-\varepsilon$ for any fixed $\varepsilon > 0$

Current best hardness of approximation: unless P=NP no 1.36 approximation for VC. Based on intricate PCP reductions

Set Cover

$\mathcal{U}$: set of n elements

$S_1, S_2, \ldots, S_m$ subsets of $\mathcal{U}$

c(i) : cost of S_i

Goal: min cost collection of sets which cover all elements (union of sets in collection is $\mathcal{U}$)

Note: Vertex Cover is a special case with each element (edge) contained in at most 2 sets (vertices)

IP/LP for Set Cover

$x(i)$: binary variable, 1 if S_i is picked in cover, 0 if S_i is not in cover

min $\sum_{i=1}^{m} c(i)\, x(i)$

s.t

$\sum_{i:\, e \in S_i} x(i) \geq 1$ for each $e \in \mathcal{U}$

$x(i) \in \{0,1\}$ for $1 \leq i \leq m$

IP/LP for Set Cover

LP

min $\sum_{i=1}^{m} c(i)\, x(i)$

s.t

$\sum_{i:\, e \in S_i} x(i) \geq 1$ for each $e \in \mathcal{U}$

$x(i) \geq 0$ for $1 \leq i \leq m$

Note: the constraint $x(i) \leq 1$ is redundant (helps simplify the dual to omit this constraint)

Rounding for Set Cover

Let f the maximum number of sets that contain any element

Note that in Vertex Cover $f = 2$

Similar to VC we can round an optimum solution x^* by picking all sets S_i with $x^*(i) \geq 1/f$

Exercise: prove that above gives an f-approx.

A different rounding

Let x^* be an optimum solution to LP

Pick all set S_i s.t $x^*(i) > 0$

Exercise: prove that this also yields an f-approx

Hint: use the dual and complementary slackness

Requires that x^* is an optimum solution unlike previous rounding

Randomized Rounding for Set Cover

If f is large previous rounding does not yield a good approximation. For example f could be as large as m!

Randomized Rounding algorithm:

Solve LP, let x^* be an optimum solution

For $i = 1$ to $c \log n$ do

Pick each set S_i in the cover independently with probability $x^*(i)$

Output all sets that are chosen in some iteration

Analysis of algorithm

What is the probability that an element e will NOT be covered in a particular iteration?

The probability that it won't be covered is exactly equal to $\prod_{i:\, e \in S_i} (1 - x^*(i))$
$\leq \prod_{i:\, e \in S_i} e^{-x^*(i)} \leq 1/e$ since $\sum_{i:\, e \in S_i} x^*(i) \geq 1$

Therefore the probability that e will not be covered after all iterations is less than $e^{-c \log n} \leq 1/n^c$

Analysis of algorithm

Therefore the probability that e will not be covered after all iterations is less than $e^{-c \log n} \leq 1/n^c$

The probability that some element is not covered is

$\leq n (1/n^c) \leq 1/n^{c-1}$

Analysis of algorithm

We now analyze the cost of the solution.

In each iteration the *expected cost* of the solution is exactly equal to $\sum_{i=1}^{m} c(i)\, x^{*}(i) = OPT_{LP}$

The total expected cost in all iterations, by linearity of expecatation, is $\leq c \log n\ OPT_{LP}$

By Markov's inequality the probability that the cost of the solution is $> 2\, c \log n\ OPT_{LP}$

is less than $1/2$

Analysis of the algorithm

Probability that all elements are not covered is $\leq 1/n^{c-1}$

Probability that cost of solution $> 2\ c \log n\ OPT_{LP} \leq 1/2$

Therefore with probability $1 - 1/2 - 1/n^{c-1}$ all elements are covered *and* cost of solution is less than $2\ c \log n\ OPT_{LP}$

Comments

- Choosing c = 2, with close to 1/2 probability we get an O(log n) approximation
- Proves that LP integrality gap is O(log n)
- Can check if solution after rounding satisfies the desired properties (cover, cost at most 2 c log n OPT_{LP}). Repeat rounding. Expected number of iteration to succed is constant. Can use Chernoff bounds (large deviation bounds) to show that a single rounding succeeds with high probability (probability at least 1 - 1/poly(n))
- Can also derandomize algorithm

Integrality gap of $\Omega(\log n)$

See an example in Vazirani's book

A different example below.

n sets and n elements

each element picks c log n sets independently to belong to

Thus the set cover instance is obtained probabilisitically

Claim: with high probability $OPT_{LP} = O(n/\log n)$

(fractional solution assigns $x(i) = \Theta(1/\log n)$ for each set)

Claim: with high probability $OPT = \Omega(n)$

Note that any feasible solution is a constant factor approximation for this instance