Approximation algorithms

Freely using Vazirani's book

Péter Gács

Computer Science Department
Boston University

Fall 06

What the course is not

This is not about "heuristic algorithms", and report on their performance in practice. It is about algorithms for which exact results are available. The "approximation" in the title just opens the range of available algorithms much wider than when we insist on exact solutions. The solution given by the algorithms will in general not be exact, but the analysis still will be.

Plan:

- Some examples
- Duality of linear programming
- Many examples using linear programming
- Other topics as time permits:
  - Shortest vector
  - Network reliability
  - Hardness of approximation
  - Other special topics by demand.

I will start with the examples and the explanation of duality. Then some of the topics will be given to student lecturers.

Grading

- Student lecture. Some of the lectures will be given by students.
- Problem solving. I will assign problems, but only nontrivial ones. Solutions will not be graded, but I will ask people to present solutions in class.
- Paper report.
- Possibly scribing, if there will be lectures on topics not following the book.

The grade will be given using informal judgement, not a formula.

Approximations

The setting

In case of NP-complete problems, maybe something can be said about how well we can approximate a solution. We will formulate the question only for problems, where we maximize (minimize) a positive, polynomial-time computable function. For object function $f(x,y)$ for $x,y \quad \{0,1\}^n$, the optimum is

$$M(x) = \max_y f(x,y)$$

where y runs over the possible "witnesses".
For $1 < \beta$, an algorithm $A(x)$ is a λ-approximation if

$$f(x,A(x)) > M(x)/\lambda.$$

For minimization problems, with minimum $m(x)$, we require $f(x,A(x)) < m(x)\lambda$.

Local search

Try local improvements as long as you can.

Example 1 (Maximum cut)

Each edge has unit weight.
Repeat: find a point on one side of the cut whose moving to the other side increases the cutsize.

Theorem 2

If you cannot improve anymore with this algorithm then you are within a factor 2 of the optimum.

Proof.

The unimprovable cut contains at least half of all edges. □

Greedy algorithms

"Build up your object" by always improving the objective function.

Example 3 (Maximum cut with weights)

Suppose that edges have weights and we are looking for the maximum-weight cut. The local search might take exponential time. But we can build up the two sides of the cut, adding points to them one-by-one, this also gives a factor 2.

We will see that "semidefinite programming" gives a better approximation.

Less greed is sometimes better

What does the greedy algorithm for vertex cover say?
The following, less greedy algorithm has better performance guarantee.
Lower bound for vertex cover is given by any maximal matching, since any optimum vertex cover must contain half of them. The following algorithm just finds a maximal matching:

```
Approx_Vertex_Cover(G)
    C ← ∅
    E ← E[G]
    while E = ∅ do
        let (u,v) be an arbitrary edge in E
        C ← C ∪ {u,v}
        remove from E every edge incident on either u or v
    return C
```

Analysis of vertex cover

1. Can the matching lower bound be used to get a better approximation?
2. Any other method leading to better approximation guarantee?

Counterexample to 1: K_n for odd n.
Question 2 is a major open problem.

Approximation problems are not invariant

to simple transformations.
Min vertex cover is equivalent to max independent set, but the latter is inapproximable (see later).
Approximating k is possible, approximating $n - k$ is not.

Bipartite graphs

Theorem 4 (König-Egerváry)

In a bipartite graph, the size of the smallest vertex cover is equal to the the size of the maximum matching.

This follows rather easily from the max-flow min-cut theorem discussed later.
This relation shows that for bipartite graphs, the minimum vertex cover question (and the maximum matching problem) is in $\mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$. (We say the problem is well-characterized).
Max-flow theory also gives us a polynomial algorithm for computing the optimum.

In general graphs, the matching lower bound on vertex cover may reach a factor 2. Example: the complete graph with an odd number of elements. In general graphs, the maximum matching problem (but probably not the vertex cover problem) is also well-characterized, it even has a polynomial solution. We will skip this.

A general strategy used in this course: transforming the problem into an integer linear programming problem.

Example 5

Vertex cover problem for $G = (V, E)$, with weigh w_i in vertex i.

Introduce variables x_i for vertex i where $x_i = 1$ if vertex x is selected, 0 otherwise.

Linear programming problem without the integrality condition:

$$\begin{array}{lll} \text{minimize} & \boldsymbol{w}^T\boldsymbol{x} & \\ \text{subject to} & x_i + x_j \geqslant 1, & (i,j) \quad E, \\ & \boldsymbol{x} \geqslant \boldsymbol{0}. & \end{array}$$

Let the optimal solution be $\boldsymbol{x}$.
Solving the integer programming problem by rounding: Choose $\bar{x}_i = 1$ if $x_i \geqslant 1/2$ and 0 otherwise.

Claim 6

Solution $\bar{\boldsymbol{x}}$ has approximation ratio 2.

Proof.

We increased each x_i by at most a factor of 2. □

The set-covering problem

Given (X, F): a set X and a family F of subsets of X, find a min-size subset of F covering X.
Example: Smallest committee with people covering all skills.
Generalization: Set S has weight $w(S) > 0$. We want a minimum-weight set cover.

```
Greedy_Set_Cover(X, F)
    U ← X
    C ← ∅
    while U = ∅ do
        select an S ∈ F that maximizes |S ∩ U|/w(S)
        U ← U ∖ S
        C ← C ∪ {S}
    return C
```

If element e was covered by set S then let $\text{price}(e) = \frac{w(S)}{S \cap U}$. Then we cover each element at minimum price (at the moment).
Note that the total final weight is $\sum_{k=1}^{n} \text{price}(e_k)$.

Analysis

Let $H(n) = 1 + 1/2 + \cdots + 1/n (\approx \ln n)$.

Theorem 7

`Greedy_Set_Cover` has a ratio bound $\max_{S \in F} H(|S|)$.

Lemma 8

For all S in F we have $\sum_{e \ S} \text{price}(e) \leqslant w(S)H(|S|)$.

Proof.

Let $e \quad S \cap S_i \setminus \bigcup_{j<i} S_j$, and $V_i = S \setminus \bigcup_{j<i} S_j$ be the remaining part of S before being covered in the greedy cover. By the greedy property,

$$\text{price}(e) \leqslant w(S)/|V_i|.$$

Let $e_1, \ldots, e_{|S|}$ be a list of elements in the order in which they are covered (ties are broken arbitrarily). Then the above inequality implies

$$\text{price}(e_k) \leqslant \frac{w(S)}{|S| - k + 1}.$$

Summing for all k proves the lemma. □

Proof of the theorem.

Let C^* be the optimal set cover and C the cover returned by the algorithm.

$$\sum_e \text{price}(e) \leqslant \sum_{S \in C} \sum_{e \in S} \text{price}(e) \leqslant \sum_{S \in C} w(S) H(|S|) \leqslant H(|S^*|) \sum_{S \in C^*} w(S)$$

where S^* is the largest set. □

Is this the best possible factor for set cover?

The answer is not known.

An alternative analysis

The result is interesting enough to deserve another analysis.
Simple fact:

Lemma 9

If $a_i, b_i > 0, i = 1, \ldots, n$ *then* $\min_i \frac{a_i}{b_i} \leqslant \frac{\sum_i a_i}{\sum_i b_i}$.

Using part of the analysis as before, if S_k is the kth set of the greedy cover, n_k the number of elements it covers newly, and V_k the part of X still to be covered, then the lemma and the greedy property implies

$$\frac{w(S_k)}{n_k} \leqslant \frac{\mathrm{OPT}}{|V_k|}.$$

Hence, using $V_1 = X$ and assuming that the greedy cover has n sets:

$$
\begin{aligned}
n_k &\geqslant \frac{w(S_k)}{\mathrm{OPT}}|V_k|, \\
|V_{k+1}| &\leqslant (1 - \frac{w(S_k)}{\mathrm{OPT}})|V_k| \leqslant |V_k| e^{-\frac{w(S_k)}{\mathrm{OPT}}}, \\
|V_n| &\leqslant |X| e^{-\frac{\sum_{i<n} w(S_i)}{\mathrm{OPT}}}, \\
\sum_{i<n} w(S_i) &\leqslant \mathrm{OPT}(\ln|X| - \ln|V_n|), \\
\sum_{i\leqslant n} w(S_i)/\mathrm{OPT} &\leqslant \ln|X| + (w(S_n)/\mathrm{OPT} - \ln|V_n|).
\end{aligned}
$$

So this analysis "almost" gives the $\ln|X|$ factor: the last term spoils it somewhat, (but for example not much if all weights are 1).

Approximation scheme

An algorithm that for every ε, gives an $(1+\varepsilon)$-approximation.

- A problem is fully approximable if it has a polynomial-time approximation scheme.
  Example: see a version KNAPSACK below.
- It is partly approximable if there is a lower bound $\lambda_{\min} > 1$ on the achievable approximation ratio.
  Example: MAXIMUM CUT, VERTEX COVER, MAX-SAT.
- It is inapproximable if even this cannot be achieved.
  Example: INDEPENDENT SET (deep result). The approximation status of this problem is different from VERTEX COVER, despite the close equivalence between the two problems.

Fully approximable version of knapsack

Given: integers $b \geqslant a_1, \ldots, a_n$, and integer weights $w_1 \geqslant \ldots \geqslant w_n$.

$$\begin{array}{ll} \text{maximize} & \boldsymbol{w}^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}^T \boldsymbol{x} \leqslant \quad b, \\ & \quad x_i = 0, 1, \ i = 1, \ldots, n. \end{array}$$

Dynamic programming: For $1 \leqslant k \leqslant n$,

$$A_k(p) = \min\{\, \boldsymbol{a}^T\boldsymbol{x} : \boldsymbol{w}^T\boldsymbol{x} = p,\ x_{k+1} = \cdots = x_n = 0 \,\}.$$

If the set is empty the minimum is ∞. Let $w = w_1 + \cdots + w_n$. The vector $(A_{k+1}(0), \ldots, A_{k+1}(w))$ can be computed by a simple recursion from $(A_k(0), \ldots, A_k(w))$. Namely, if $w_{k+1} > p$ then $A_{k+1}(p) = A_k(p)$. Otherwise,

$$A_{k+1}(p) = \min\{\, A_k(p), w_{k+1} + A_k(p - w_{k+1}) \,\}.$$

The optimum is $\max\{\, p : A_n(p) \leqslant b \,\}$.
Complexity: roughly $O(nw)$ steps.
Why is this not a polynomial algorithm?

Idea for approximation: break each w_i into a smaller number of big chunks, and use dynamic programming. Let $r > 0$, $w_i = w_i/r$.

$$\begin{array}{ll} \text{maximize} & (\boldsymbol{w}\ )^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}^T\boldsymbol{x} \leqslant\ \ b, \\ & x_i = 0, 1,\ i = 1, \ldots, n. \end{array}$$

For the optimal solution $\boldsymbol{x}$ of the changed problem, estimate $\mathrm{OPT}/\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{w}^T\boldsymbol{x}/\boldsymbol{w}^T\boldsymbol{x}$. We have

$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{x}/r &\geqslant (\boldsymbol{w})^T\boldsymbol{x} \geqslant (\boldsymbol{w})^T\boldsymbol{x} \geqslant (\boldsymbol{w}/r)^T\boldsymbol{x} - n,\\ \boldsymbol{w}^T\boldsymbol{x} &\geqslant \mathrm{OPT} - r\cdot n.\end{aligned}$$

Let $r = \varepsilon w_1/n$, then

$$\frac{(\boldsymbol{w})^T\boldsymbol{x}}{\mathrm{OPT}} \geqslant 1 - \frac{\varepsilon w_1}{\mathrm{OPT}} \geqslant 1-\varepsilon.$$

With $w = \sum_i w_i$, the amount of time is of the order of

$$nw/r = n^2 w/(w_1\varepsilon) \leqslant n^3/\varepsilon,$$

which is polynomial in $n, (1/\varepsilon)$.

Look at the special case of knapsack, with $w_i = a_i$. Here, we just want to fill up the knapsack as much as we can. This is equivalent to minimizing the remainder,

$$b - \sum_i \boldsymbol{a}^T \boldsymbol{x}.$$

But this minimization problem is inapproximable.

The traveling salesman problem

Given a complete graph with nonnegative edge costs $\text{cost}(u, v)$, find a minimum-cost cycle visiting every vertex exactly once.
Why cycle? If it is cheaper for the salesman to use some nodes twice, why not allow it? Using this observation leads to the metric TSP problem, in which

$$\text{cost}(u, w) \leqslant \text{cost}(u, v) + \text{cost}(v, w).$$

It is easy to see that

- The non-metric TSP problem is not approximable. Indeed, the Hamilton cycle problem is reducible to any attempted approximation.
- The metric TSP problem is the same as allowing a tour to visit nodes twice. It is approximable within a factor of 2: just go around an optimal spanning tree.

The approximation factor can be improved to 3/2: see the book.

The Steiner tree problem

Graph $G = (V, E)$, with edge cost $\text{cost}(u, v)$, required subset $R \quad V$. Find a set of edges in E that connect all points of R. These edges necessarily form a tree. The vertices additional to R in these edges are called the Steiner vertices.

Observation: the metric version has the same approximation factor. So, let us require $\text{cost}(u, v)$ to be a metric.

Factor 2 approximation: minimum spanning tree over R.

Proof.

Steiner tree $\rightarrow$ Euler cycle $\rightarrow$ Hamilton cycle on R $\rightarrow$ spanning tree of R. $\square$

The minimum cut problem

Given a graph $G = (V, E)$ and nonnegative weight function $w(e)$ on the edges.
Cut. Weight of the cut.
The $s - t$ cut problem: find a minimum-weight cut separating s and t.
The maximum flow algorithm gives a polynomial-time solution. See later, at the treatment of linear programming duality.

The multiway cut problem

Given points $s_1, \ldots, s_k$, a multiway cut is a set of edges disconnecting all s_i from each other. NP-hard. We will give a very simple approximation algorithm. The (clever) analysis will show that its factor is $2(1 - 1/k)$.

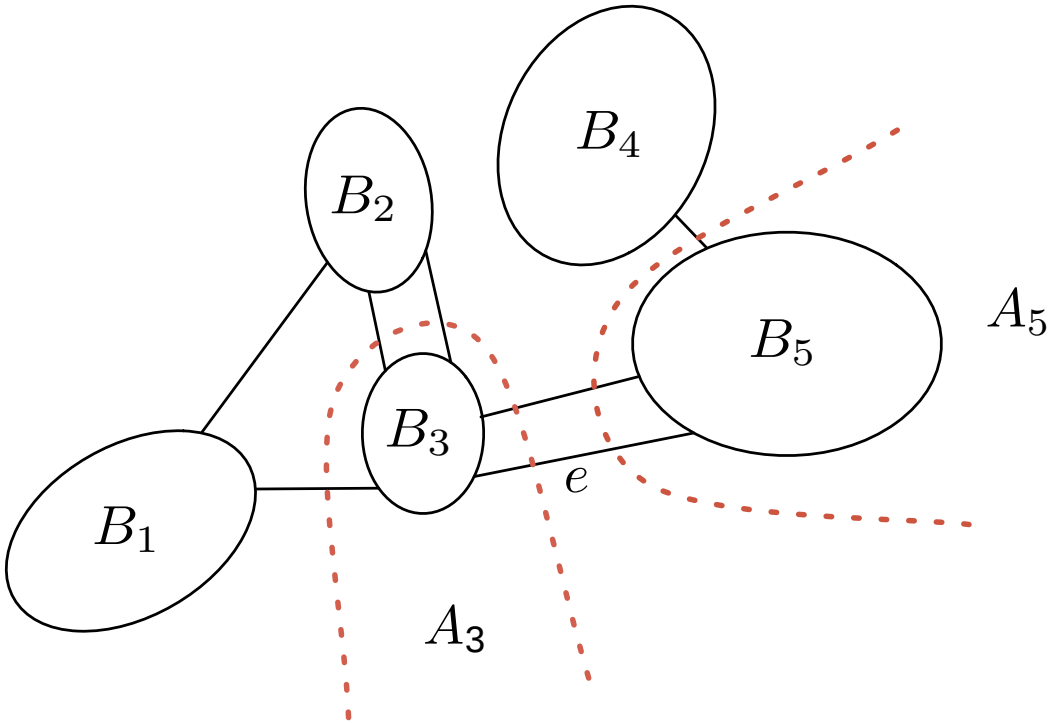
Isolating cut: for some i, separates s_i from the rest.

Approximation algorithm:

1. For each $i = 1, \ldots, k$, compute a minimum-weight isolating cut C_i for s_i.
2. Output the union C of the $(k - 1)$ lightest ones.

Theorem 10

This gives an approximation factor $2(1 - 1/k)$.



For the proof, let B_i be the component of s_i in some optimal multiway cut A. Let A_i be the cut separating B_i. Since each edge of A connects two components, we have $2w(A) = \sum_i w(A_i)$.

Hence

$$2w(A) = \sum_i w(A_i) \geqslant \sum_i w(C_i),$$
$$2(1-1/k)w(A) \geqslant (1-1/k)\sum_i w(C_i) \geqslant w(C)$$

since we discarded the heaviest C_i.

The k-cut problem does not sound that interesting in itself, but the Vazirani book's solution is a good opportunity to learn about Gomory-Hu trees.
But try to come up with a polynomial algorithm for 3-cuts, at least!

The (metric) k-center problem

Given a complete graph with edge lengths $d(u,v)$ that form a metric (we will say, a metric space X) and a as set S of points, let

$$d(x,S) = \min_{y \ S} d(x,y).$$

Problem 11

Find a set S of k points for which $\max_{x \ X} d(x,S)$ *is minimal.*

Covering, packing

Let

$$B(x, r)$$

be the ball of center x and radius r, that is the set of points of distance $\leqslant r$ from x. The k-center is looking for the smallest number of k balls covering V. This connects with the following classical questions for a metric space X:

- Ball covering: what is the minimum number $C(r, X)$ of r-balls covering X?
- Ball packing: what is the maximum number $P(r, X)$ of disjoint r-balls that can be packed into X?

Observation 1

$C(2r, X) \leqslant P(r, X) \leqslant C(r, X)$.

Using this for a factor 2 approximation algorithm.

If points have weights and we are looking for a set of weight $\leqslant W$, similar ideas lead to a factor 3 algorithm.

Linear programming

How about solving a system of linear inequalities?

$$\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}.$$

We will try to solve a seemingly more general problem:

$$\begin{array}{ll} \text{maximize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}. \end{array}$$

This optimization problem is called a linear program. (Not program in the computer programming sense.)

Example 12

Three voting districts: urban, suburban, rural.
Votes needed: 50,000, 100,000, 25,000.
Issues: build roads, gun control, farm subsidies, gasoline tax.
Votes gained, if you spend $ 1000 on advertising on any of these issues:

adv. spent	policy	urban	suburban	rural
x_1	build roads	-2	5	3
x_2	gun control	8	2	-5
x_3	farm subsidies	0	0	10
x_4	gasoline tax	10	0	-2
	votes needed	$50,000$	$100,000$	$25,000$

Minimize the advertising budget $(x_1 + \cdots + x_4) \cdot 1000$.

The linear programming problem:

$$\begin{array}{lrcrcrcr} \text{minimize} & x_1 & + & x_2 & + & x_3 & + & x_4 \\ \text{subject to} & -2x_1 & + & 8x_2 & & & + & 10x_4 \geqslant 50,000 \\ & 5x_1 & + & 2x_2 & & & & \geqslant 100,000 \\ & 3x_1 & - & 5x_2 & + & 10x_3 & - & 2x_4 \geqslant 25,000 \end{array}$$

Implicit inequalities: $x_i \geqslant 0$.

Two-dimensional example

$$
\begin{array}{lrcl}
\text{maximize} & x_1 + x_2 & & \\
\text{subject to} & 4x_1 - x_2 & \leqslant & 8 \\
& 2x_1 + x_2 & \leqslant & 10 \\
& 5x_1 - 2x_2 & \geqslant & -2 \\
& x_1, \quad x_2 & \geqslant & 0
\end{array}
$$

Graphical representation, see book.
Convex polyhedron, extremal points.
The simplex algorithm: moving from an extremal point to a nearby one (changing only two inequalities) in such a way that the objective function keeps increasing.

Worry: there may be too many extremal points. For example, the set of $2n$ inequalities

$$0 \leqslant x_i \leqslant 1, \qquad i = 1, \ldots, n$$

has 2^n extremal points.

Standard and slack form

Standard form

$$\begin{array}{ll} \text{maximize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \\ & \boldsymbol{x} \geqslant \boldsymbol{0} \end{array}$$

Objective function, constraints, nonnegativity constraints, feasible solution, optimal solution, optimal objective value. Unbounded: if the optimal objective value is infinite.
Converting into standard form:

$$x_j = x_j - x_j \text{ , subject to } x_j, x_j \geqslant 0.$$

Handling equality constraints.

Slack form

In the slack form, the only inequality constraints are nonnegativity constraints. For this, we introduce slack variables on the left:

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j.$$

In this form, they are also called basic variables. The objective function does not depend on the basic variables. We denote its value by z.

Example for the slack form notation:

$$\begin{aligned} z &= 2x_1 - 3x_2 + 3x_3 \\ x_4 &= 7 - x_1 - x_2 + x_3 \\ x_5 &= -7 + x_1 + x_2 - x_3 \\ x_6 &= 4 - x_1 + 2x_2 - 2x_3 \end{aligned}$$

More generally: $B =$ set of indices of basic variables, $|B| = m$.
$N =$ set of indices of nonbasic variables, $|N| = n$,
$B \quad N = \{1, \ldots, m+n\}$. The slack form is given by $(N, B, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, v)$:

$$\begin{aligned} z &= v + \sum_{j \ N} c_j x_j \\ x_i &= b_i - \sum_{j \ N} a_{ij} x_j \qquad \text{for } i \quad B. \end{aligned}$$

Note that these equations are always independent.

Maximum flow

Capacity $c(u,v) \geqslant 0$.

$$\begin{array}{lll} \text{maximize} & \sum_v f(s,v) & \\ \text{subject to} & f(u,v) \leqslant c(u,v) & \\ & f(u,v) = -f(v,u) & \\ & \sum_v f(u,v) = 0 & \text{for } u \quad V - \{s,t\} \end{array}$$

The matching problem.
Given m workers and n jobs, and a graph connecting each worker with some jobs he is capable of performing. Goal: to connect the maximum number of workers with distinct jobs.
This can be reduced to a maximum flow problem.

The simplex algorithm

Slack form.
A basic solution: set each nonbasic variable to 0. Assume that there is a basic feasible solution (see later how to find one).
Iteration step idea: try to raise the value of the objective function by changing a nonbasic variable x_j until some basic variable x_i turns 0 (its equality constraint becomes tight). Then exchange x_i and x_j.
Question: if this is not possible, are we done?

Example 13

$$\begin{aligned} z &= 3x_1 + x_2 + 2x_3 \\ x_4 &= 30 - x_1 - x_2 + 3x_3 \\ x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\ x_6 &= 36 - 4x_1 - x_2 - 2x_3 \end{aligned}$$

Since all b_i are positive, the basic solution is feasible. Increase x_1 until one of the constraints becomes tight: now, this is x_6 since b_i/a_{i1} is minimal for $i = 6$. Pivot operation:

$$x_1 = 9 - x_2/4 - x_3/2 - x_6/4$$

Here, x_1 is the entering variable, x_6 the leaving variable.

Rewrite all other equations, substituting this x_1:

$$\begin{aligned} z &= 27 + x_2/4 + x_3/2 - 3x_6/4 \\ x_1 &= 9 - x_2/4 - x_3/2 - x_6/4 \\ x_5 &= 21 - 3x_2/4 - 5x_3/2 + x_6/4 \\ x_6 &= 6 - 3x_2/2 - 4x_3 + x_6/2 \end{aligned}$$

Formal pivot algorithm: no surprise.

Lemma 14

The slack form is uniquely determined by the set of basic variables.

Proof.

Simple, using the uniqueness of linear forms. □

When can we not pivot?

- unbounded case
- optimality

The problem of cycling

It can be solved, though you will not encounter it in practice. Perturbation, or "Bland's Rule": choose variable with the smallest index. (No proof here that this terminates.) Geometric meaning: walking around a fixed extremal point, trying different edges on which we can leave it while increasing the objective.

End result, used later for duality:

Theorem 15

If there is an optimum v then there is a basis $B \quad \{1,\dots,m+n\}$ belonging to a basic feasible solution, and coefficients $\tilde{c}_i \leqslant 0$ such that

$$\boldsymbol{c}^T\boldsymbol{x} = v + \tilde{\boldsymbol{c}}^T\boldsymbol{x},$$

where $\tilde{c}_i = 0$ for $i \quad B$.

Other formulation: if the original problem has the form

$$\begin{array}{ll} \text{maximize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \\ & \boldsymbol{x} \geqslant \boldsymbol{0} \end{array}$$

with $\boldsymbol{c} \leqslant \boldsymbol{0}$, $\boldsymbol{b} \geqslant \boldsymbol{0}$ then $\boldsymbol{0}$ is obviously an optimal solution. The above theorem says that if there is an optimal solution then by introducing slack variables and changing basis we can always bring the system into such a form.

Initial basic feasible solution

(Frequently obvious from the problem definition.)
Solve the following auxiliary problem, with an additional variable x_0:

$$\begin{array}{lll} \text{minimize} & x_0 & \\ \text{subject to} & \boldsymbol{a}_i^T \boldsymbol{x} - x_0 \leqslant \boldsymbol{b} & i = 1, \ldots, m, \\ & \boldsymbol{x}, \quad x_0 \geqslant 0 & \end{array}$$

If the optimal x_0 is 0 then the optimal basic feasible solution is a basic feasible solution to the original problem.

Complexity of the simplex method

Each pivot step takes $O(mn)$ algebraic operations.
How many pivot steps? Can be exponential.
Does not occur in practice, where the number of needed iterations is rarely higher than $3\max(m,n)$. Does not occur on "random" problems, but mathematically random problems are not typical in practice.
Spielman-Teng: on a small random perturbation of a linear program (a certain version of) the simplex algorithm terminates in polynomial time (on average).
There exists also a polynomial algorithm for solving linear programs (see later), but it is rarely competitive with the simplex algorithm in practice.

Duality

Primal (standard form): maximize $\boldsymbol{c}^T\boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$ and $\boldsymbol{x} \geqslant \boldsymbol{0}$.
Value of the optimum (if feasible): z . Dual:

$$\begin{array}{rr} \boldsymbol{A}^T\boldsymbol{y} \geqslant \boldsymbol{c} & \boldsymbol{y}^T\boldsymbol{A} \geqslant \boldsymbol{c}^T \\ \boldsymbol{y} \geqslant \boldsymbol{0} & \boldsymbol{y}^T \geqslant \boldsymbol{0} \\ \min \quad \boldsymbol{b}^T\boldsymbol{y} & \min \quad \boldsymbol{y}^T\boldsymbol{b} \end{array}$$

Value of the optimum if feasible: t .

Proposition 16 (Weak duality)

$z \leqslant t$, moreover for every pair of feasible solutions $\boldsymbol{x}, \boldsymbol{y}$ of the primal and dual:

$$\boldsymbol{c}^T\boldsymbol{x} \leqslant \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{y}^T\boldsymbol{b} = \boldsymbol{b}^T\boldsymbol{y}. \tag{1}$$

Use of duality. If somebody offers you a feasible solution to the dual, you can use it to upperbound the optimum of the final (and for example decide that it is not worth continuing the simplex iterations).

Interpretation:

- b_i = the total amount of resource i that you have (kinds of workers, land, machines).
- a_{ij} = the amount of resource i needed for activity j.
- c_j = the income from a unit of activity j.
- x_j = amount of activity j.

$\boldsymbol{Ax} \leqslant \boldsymbol{b}$ says that you can use only the resources you have.
Primal problem: maximize the income $\boldsymbol{c}^T\boldsymbol{x}$ achievable with the given resources.

T

Resource i has price y_i. Total income:

$$L(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{y}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) = (\boldsymbol{c}^T - \boldsymbol{y}^T\boldsymbol{A})\boldsymbol{x} + \boldsymbol{y}^T\boldsymbol{b}.$$

Let

$$f(\hat{\boldsymbol{x}}) = \inf_{\boldsymbol{y}\geqslant\boldsymbol{0}} L(\hat{\boldsymbol{x}},\boldsymbol{y}) \leqslant L(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) \leqslant \sup_{\boldsymbol{x}\geqslant\boldsymbol{0}} L(\boldsymbol{x},\hat{\boldsymbol{y}}) = g(\hat{\boldsymbol{y}}).$$

Then $f(\boldsymbol{x}) > -\infty$ needs $\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$. Hence if the primal is feasible then for the optimal $\boldsymbol{x}$ (choosing $\boldsymbol{y}$ to make $\boldsymbol{y}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) = 0$) we have

$$\sup_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} = z \ .$$

Similarly $g(\boldsymbol{y}) < \infty$ needs $\boldsymbol{c}^T \leqslant \boldsymbol{y}^T\boldsymbol{A}$, hence if the dual is feasible then we have

$$z \leqslant \inf_{\boldsymbol{y}} g(\boldsymbol{y}) = (\boldsymbol{y})^T\boldsymbol{b} = t \ .$$

Complementary slackness conditions:

$$\boldsymbol{y}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) = \boldsymbol{0}, \quad (\boldsymbol{y}^T\boldsymbol{A} - \boldsymbol{c}^T)\boldsymbol{x} = \boldsymbol{0}.$$

Proposition 17

Equality of the primal and dual optima implies complementary slackness.

Interpretation:

- Inactive constraints have shadow price $y_i = 0$.
- Activities that do not yield the income required by shadow prices have level $x_j = 0$.

Theorem 18 (Strong duality)

The primal problem has an optimum if and only if the dual is feasible, and we have

$$z = \max \boldsymbol{c}^T \boldsymbol{x} = \min \boldsymbol{y}^T \boldsymbol{b} = t \ .$$

This surprising theorem says that there is a set of prices (called shadow prices) which will force you to use your resources optimally. Many interesting uses and interpretations, and many proofs.

Our proof of strong duality uses the following result of the analysis of the simplex algorithm.

Theorem 19

If there is an optimum v then there is a basis $B \quad \{1,\dots,m+n\}$ belonging to a basic feasible solution, and coefficients $\tilde{c}_i \leqslant 0$ such that

$$\boldsymbol{c}^T\boldsymbol{x} = v + \tilde{\boldsymbol{c}}^T\boldsymbol{x},$$

where $\tilde{c}_i = 0$ for $i \quad B$.

Define the nonnegative variables

$$\tilde{y}_i = -\tilde{c}_{n+i} \qquad i = 1,\dots,m.$$

For any x, the following transformation holds, where $i = 1, \ldots, m$, $j = 1, \ldots, n$:

$$\begin{aligned}\sum_j c_j x_j &= v + \sum_j \tilde{c}_j x_j + \sum_i \tilde{c}_{n+i} x_{n+i} \\ &= v + \sum_j \tilde{c}_j x_j + \sum_i (-\tilde{y}_i)(b_i - \sum_j a_{ij} x_j) \\ &= v - \sum_i b_i \tilde{y}_i + \sum_j (\tilde{c}_j + \sum_i a_{ij} \tilde{y}_i) x_j.\end{aligned}$$

This is an identity for x, so $v = \sum_i b_i \tilde{y}_i$, and also $c_j = \tilde{c}_j + \sum_i a_{ij} \tilde{y}_i$. Optimality implies $\tilde{c}_j \leqslant 0$, which implies that $\tilde{y}_i$ is a feasible solution of the dual.

Linear programming and linear inequalities

Any feasible solution of the set of inequalities

$$\begin{aligned} Ax &\leqslant b \\ A^T y &\geqslant c \\ c^T x - b^T y &= 0 \\ x, y &\geqslant 0 \end{aligned}$$

gives an optimal solution to the original linear programming problem.

Theory of alternatives

Theorem 20 (Farkas Lemma, not as in the book)

A set of inequalities $\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$ is unsolvable if and only if a positive linear combination gives a contradiction: there is a solution $\boldsymbol{y} \geqslant \boldsymbol{0}$ to the inequalities

$$\boldsymbol{y}^T \boldsymbol{A} = 0,$$
$$\boldsymbol{y}^T \boldsymbol{b} < 0.$$

For proof, translate the problem to finding an initial feasible solution to standard linear programming.

We use the homework allowing variables without nonnegativity constraints:

$$\begin{array}{ll} \text{maximize} & z \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} + z \cdot \boldsymbol{e} \leqslant \boldsymbol{b} \end{array} \tag{2}$$

Here, $\boldsymbol{e}$ is the vector consisting of all 1's. The dual is

$$\begin{array}{ll} \text{minimize} & \boldsymbol{y}^T \boldsymbol{b} \\ \text{subject to} & \boldsymbol{y}^T \boldsymbol{A} = \boldsymbol{0} \\ & \boldsymbol{y}^T \boldsymbol{e} = 1 \\ & \boldsymbol{y}^T \geqslant \boldsymbol{0} \end{array} \tag{3}$$

The original problem has no feasible solution if and only if $\max z < 0$ in (2). In this case, $\min \boldsymbol{y}^T \boldsymbol{b} < 0$ in (3). (Condition $\boldsymbol{y}^T \boldsymbol{e} = 1$ is not needed.)

Dual for max-flow: min-cut

Directed complete graph $G = (V, E)$. Edge (u, v) has capacity $c(u, v)$. Looking for nonnegative flow of value $f(u, v)$ on edge (u, v).

$$\begin{array}{lrcll} \text{maximize} & \sum_{v \ V} f(s, v) & & & \\ \text{subject to} & f(u, v) & \leqslant & c(u, v), & u, v \quad V, \\ & \sum_{v \ V} f(u, v) - f(v, u) & = & 0, & u \quad V \setminus \{s, t\}, \\ & f(u, v) & \geqslant & 0, & u, v \quad V. \end{array}$$

A dual variable for each constraint. For $f(u, v) \leqslant c(u, v)$, call it $y(u, v)$, for

$$\sum_{v \ V} f(u, v) - f(v, u) = 0$$

call it $y(u)$ (this is not restricted by sign).

Dual constraint for each primal variable $f(u,v)$. If $u,v = s$ then $f(u,v)$ has coefficient 0 in the objective function. The dual inequality for equation for $u = s, v = t$ is $y(u,v) + y(u) - y(v) \geqslant 0$, or

$$y(u,v) \geqslant y(v) - y(u).$$

For $u = s, v = t$: $y(s,v) - y(v) \geqslant 1$, or $y(s,v) \geqslant y(v) - (-1)$.
For $u = s$ but $v = t$ we have $y(u,t) + y(u) = 0$, or $y(u,t) \geqslant 0 - y(u)$.
For $u = s, v = t$: $y(s,t) \geqslant 1$, or $y(s,t) \geqslant 0 - (-1)$.
Setting $y(s) = -1, y(t) = 0$, all these inequalities can be summarized in $y(u,v) \geqslant y(v) - y(u)$ for all u, v.

The objective function is $\sum_{u,v} c(u,v)y(u,v)$. Of course, it is smallest by setting $y(u,v) = |y(v) - y(u)|^+$. Simplified dual problem:

$$\begin{array}{ll} \text{minimize} & \sum_{u,v} c(u,v)|y(v) - y(u)|^+ \\ \text{subject to} & y(s) = -1, \quad y(t) = 0. \end{array}$$

Let us require $y(s) = 0, y(t) = 1$ instead; the problem remains the same.

Claim 21

There is an optimal solution in which each $y(u)$ is 0 or 1.

Proof.

Assume that there is a $y(u)$ that is not 0 or 1. If it is outside the interval $[0,1]$ then moving it towards this interval decreases the objective function, so assume they are all inside. If there are some variables $y(u)$ inside this interval then move them all by the same amount either up or down until one of them hits 0 or 1. One of these two possible moves will not increase the objective function. Repeat these actions until each $y(u)$ is 0 or 1. □

Let $\boldsymbol{y}$ be an optimal solution in which each $y(u)$ is either 0 or 1. Let

$$S = \{\, u : y(u) = 0 \,\}, \quad T = \{\, u : y(u) = 1 \,\}.$$

Then $s \quad S$, $t \quad T$. The objective function is

$$\sum_{u \quad S, v \quad T} c(u, v).$$

This is the value of the "cut" (S, T). So the dual problem is about finding a minimum cut, and the duality theorem implies the max-flow/min-cut theorem.

Alternative treatment of max flow-min cut

Let P be the set of all $s-t$ paths. Given a path p and a real number $\alpha \geqslant 0$, we can define a flow $f(u,v)$ in which $f(u,v) = \alpha$ for an edge (u,v) in the path, $f(u,v) = -\alpha$ for each edge (v,u) on the path, and $f(u,v) = 0$ otherwise. Let us call such a flow a path flow defined by (p,α). It is easy to see that each $s-t$ flow is the sum of a finite number of path-flows (at most as many as the number of edges).

Let us assign a variable f_p for each path p P. (This is exponentially many variables, but in theory, we can do this.)

Assigning values $f_p \geqslant 0$ defines a flow which is the sum of all the path flows defined by (p,f_p), p P.

New formulation of the max flow problem:

$$\begin{array}{lll} \text{maximize} & \sum_{p \ P} f_p & \\ \text{subject to} & \sum_{p:e \ p} f_p \leqslant c_e & e \quad E, \\ & f_p \geqslant 0 & p \quad P. \end{array}$$

Dual: a variable d_e for each edge constraint. An inequality for each path:

$$\begin{array}{lll} \text{minimize} & \sum_{e \ E} c_e d_e & \\ \text{subject to} & \sum_{e \ p} d_e \geqslant 1 & p \quad P, \\ & d_e \geqslant 0 & e \quad E. \end{array}$$

The distance function d_e can be called a fractional cut. The shortest-path algorithm generates from it a cut, but the relation to min cut is not clear.

Maximum bipartite matching

Bipartite graph with left set A, right set B and edges $E \quad A \times B$.
Interpretation: elements of A are workers, elements of B are jobs.
$(a, b) \quad E$ means that worker a has the skill to perform job b. Two edges are disjoint if both of their endpoints differ. Matching: a set M of disjoint edges. Maximum matching: a maximum-size assignment of workers to jobs.
Covering set $C \quad A \quad B$: a set with the property that for each edge $(a, b) \quad E$ we have $a \quad C$ or $b \quad C$.
Clearly, the size of each matching is $\leqslant$ the size of each covering set.

Theorem 22

The size of a maximum matching is equal to the size of a minimum covering set.

There is a proof by reduction to the flow problem and using the max-flow min-cut theorem.

Ellipsoid algorithm

We will only solve the feasibility problem for a set of linear inequalities for $\boldsymbol{x} = (x_1, \ldots, x_n)^T$:

$$\boldsymbol{a_i}^T \boldsymbol{x} \leqslant b_i, \qquad i = 1, \ldots, m$$

with the property that the set of solutions is bounded and has an interior, namely there are points satisfying all inequalities strictly. The other cases can be reduced to this easily.
Assumption: that the input consists of rational numbers with length $\leqslant L$.

The algorithm

1. Find a ball E_1 with the property that if there is a solution then B contains one. In the following steps, E_k is an ellipsoid (ball is a special case).
2. Repeat the following.
   - Check if the center of E_k is a solution. If not, find the violated inequality $\boldsymbol{a_i}^T \boldsymbol{x} \leqslant b_i$.
   - Find the ellipsoid E_{k+1} with the smallest volume enclosing the half-ellipsoid

$$\{\boldsymbol{x} \quad E_k : \boldsymbol{a_i}^T \boldsymbol{x} \leqslant b_i\}.$$

Proposition 23 (can be checked)

$\mathrm{vol}(E_{k+1})/\mathrm{vol}(E_k) \leqslant e^{-1/(2n+1)}$.

Thus, if the solution has not been found in k steps, then the volume of the set of solutions is $\leqslant e^{-k/(2n+1)}\mathrm{vol}(E_1)$.

Proposition 24 (can be checked)

$\mathrm{vol}(E_1) \leqslant 2^{n^2 L}$.

Proposition 25

If the set of solutions is not empty then its volume is at least 2^{-2n^2L}.

Proof sketch.

The set of solutions contains a simplex formed by $(n+1)$ extremal points. Its volume can be expressed with the help of certain integer determinants, and the ones in the denominator can be upperbounded. □

The above three statements imply that, for a certain constant c, if ellipsoid algorithm does not terminate in cn^4 iterations then it there is no solution.

Oracle formulation

The ellipsoid algorithm can be generalized to the case when the inequalities $\boldsymbol{a}_i^T \boldsymbol{x} \leqslant b_i$ are not given explicitly. Rather, some polynomial-timte oracle algorithm is given that for every $\boldsymbol{x}$ that is not a solution returns a hyperplane separating $\boldsymbol{x}$ from the set of solutions.

Example 26

In the above alternative formulation of the fractional min-cut problem, there were as many constraints as there are paths. But for every vector $\boldsymbol{d}$ that is not a solution, Dijkstra's algorithm produces a shortest $s - t$ path, and its constraint will be violated.

Probabilistic rounding

Take a 0-1-valued integer programming problem. Solve the fractional relaxacion getting values x_i. Consider these probabilities and set $x_i = 1$ with probability x_i independently for each i. Analyze what you get.

Example 27

Set cover, covering an n-element set. Let x_S be the selection variable of set S. If element e is in sets $S_1, \ldots, S_k$, then the probability of not covering it is

$$p = (1 - x_{S_1}) \cdots (1 - x_{S_k}).$$

We know $x_{S_1} + \cdots + x_{S_k} \geqslant 1$. From this it follows by the arithmetic-geometric mean inequality that

$$p \leqslant (1 - 1/k)^k \leqslant e^{k(-1/k)} = 1/e.$$

Repeating the rounding $d \ln n$ times (always adding the newly selected sets) the probability of not covering an element will be decreased to n^{-d}, so the probability of not covering something is at most n^{1-d}.

Linear programming for set cover

Let for set S be $p_S = 1$ if S is selected and 0 otherwise. Set cover problem, without integrality condition:

$$\begin{array}{ll} \text{minimize} & \sum_S w_S p_S \\ \text{subject to} & \sum_{S \ \ x} p_S \geqslant 1, \ x \quad X, \\ & p_S \geqslant 0, \ S \quad F, \end{array}$$

Dual with variables c_x, $x \quad X$:

$$\begin{array}{ll} \text{maximize} & \sum_{x \ \ X} c_x \\ \text{subject to} & \sum_{x \ \ S} c_x \leqslant w_S, \ S \quad F, \\ & c_x \geqslant \ 0, \ x \quad X. \end{array}$$

Interpretation: packing amounts c_x of something into the elements, where the total amount in each set S is limited by w_S. The maximum allowable total amount packed turns out to be the value of the minimum fractional set cover. In general, talking about covering LP and packing LP.

Primal-dual schema

Let us generalize the above problem. Primal (a "covering problem"):

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \geqslant b_i, \ i = 1, \ldots, m, \\ & x_j \geqslant 0, \ \ j = 1, \ldots, n. \end{array}$$

Dual (a "packing problem"):

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} b_i y_i \\ \text{subject to} & \sum_{i=1}^{n} a_{ij} y_i \leqslant c_j, \ \ j = 1, \ldots, n. \\ & y_i \geqslant 0, \ i = 1, \ldots, m. \end{array}$$

For some $\alpha, \beta \geqslant 1$, formally relax the complementary slackness:
Primal conditions: $x_j > 0 \qquad c_j/\alpha \leqslant \sum_j a_{ij} y_i \leqslant c_j$.
Dual conditions: $y_i > 0 \qquad b_i \leqslant \sum_j a_{ij} x_j \leqslant \beta b_i$.

Proposition 28

If the primal and dual feasible solutions satisfy these conditions, then

$$\boldsymbol{c}^T \boldsymbol{x} \leqslant \alpha\beta \boldsymbol{b}^T \boldsymbol{y}.$$

Proof straightforward.

The primal-dual schema

- Start from an infeasible integer primal and a feasible dual (typically $\boldsymbol{x} = \mathbf{0}, \boldsymbol{y} = \mathbf{0}$).
- Keep improving the feasibility of the primal, keeping it integral, and the optimality of the dual.
  The primal guides the improvements of the dual and vice versa.

Application to set cover

Set cover problem, without integrality condition. Set system $\mathcal{S}$, universe U.

$$\begin{array}{ll} \text{minimize} & \sum_S c(S)x_S \\ \text{subject to} & \sum_{S \ni e} x_S \geqslant 1, \ e \in U, \\ & x_S \geqslant 0, \ S \in \mathcal{S}, \end{array}$$

Dual with variables y_e, $e \in U$:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in U} y_e \\ \text{subject to} & \sum_{e \in S} y_e \leqslant c(S), \ S \in \mathcal{S}, \\ & y_x \geqslant 0, \ e \in U. \end{array}$$

Primal complementary slackness conditions for each S, with factor $\alpha = 1$: $x_S = 0 \quad \sum_{e \in S} y_e = c(S)$.
Set S is tight when this holds. Plan: use only tight sets.

Relaxed dual complementary slackness conditions for each e, with factor $\beta = f$:

$$y_e = 0 \qquad \sum_{S \ni e} x_S \leqslant f.$$

This is satisfied automatically.

The algorithm

1. Start with $\boldsymbol{x} = \mathbf{0}, \boldsymbol{y} = \mathbf{0}$.
2. Repeat, until all elements are covered:
   Pick uncovered element e, raise y_e until some set S goes tight. Add S to the set cover.

Since the relaxed complementary slackness conditions hold at the end, we achieved the approximation factor $\alpha\beta = 1$.

Simplicity

As opposed to the simplex method, the successive improvements were not accompanied by any linear transformations accompanying a basis change.

Sparsest cut

Demands multicommodity flow

Undirected graph $G = (V, E)$. Capacity c_e of edge e.
Pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$. Demand $\mathrm{dem}(i)$ of commodity i.
Maximize f where $f \cdot \mathrm{dem}(i)$ will flow from s_i to t_i.
For upper bounds, for a cut $(S, \overline{S})$ define

$$\mathrm{dem}(S) = \sum_{i: s_i, t_i \text{ are separated by } S} dem(i).$$

The sparsity of the cut is

$$\frac{c(S)}{\mathrm{dem}(S)}.$$

This is an upper bound on f.

Question 1

How tight a bound is given by the sparsest cut? How to compute a sparse cut?

Let P_i be the set of $s_i - t_i$ paths.

$$\begin{array}{lrll} \text{maximize} & f & & \\ \text{subject to} & \sum_{p \in P_i} f_p^i \geq & f \cdot \mathrm{dem}(i), & i = 1, \ldots, k, \\ & \sum_{i=1}^{k} \sum_{p \in P_i : e \in p} f_p^i \leq & c_e, & e \in E, \\ & f \geq & 0, & \\ & f_p^i \geq & 0. & \end{array}$$

Dual: a variable d_e for each edge constraint, variable l_i for each demand.

$$\begin{array}{lrll} \text{minimize} & \sum_{e \in E} c_e d_e & & \\ \text{subject to} & \sum_{e \in p} d_e \geq & l_i, & p \in P_i,\ i = 1, \ldots, k, \\ & \sum_{i=1}^{k} l_i \mathrm{dem}(i) \geq & 1, & i = 1, \ldots, k, \\ & d_e \geq & 0, & \\ & l_i \geq & 0. & \end{array}$$

Observation 2

The optimum of the dual can be achieved using

- *A metric* d_e.
- $l_i = d_{(s_i,t_i)}$.
- *Requiring* $\sum_i d_{(s_i,t_i)} \mathrm{dem}(i) = 1$.

This implies the following, with H the graph with edges (s_i, t_i):

Theorem 29

$$f = \min_{\text{metric } \boldsymbol{d}} \frac{\sum_e c_e d_e}{\sum_{e \ H} \mathrm{dem}(e) d_e}$$

The connection with cuts

For a cut S, let $\delta(S)$ be the set of edges across it. A function

$$y : 2^V \to \mathbb{R}_+$$

is called a cut measure. Each cut measure defines a metric D^y:

$$D^y_e = \sum_{S \ e} y_S.$$

A cut measure y is a cut packing for metric d if $D^y_e \leqslant d_e$ for all e. It is exact if equality. For $\beta \geqslant 1$ it is β-approximate if $d_e \leqslant \beta \cdot D^y_e$ for all e.

Theorem 30

Let $\mathbf{d}$ be a metric obtained from the optimal dual of the demand multicommodity flow problem. If there is a β-approximate cut packing for $\mathbf{d}$ then the sparsity of the sparsest cut is $\leqslant \beta \cdot f$.

l_1 metric

The metrics that have exact cut packings have a common origin.
A norm $\|x\|$ over a vector space:

$$\begin{aligned} \|x\| = 0 &\Leftrightarrow x = 0, \\ \|\lambda x\| &= |\lambda| \cdot \|x\|, \\ \|x+y\| &\leqslant \|x\| + \|y\|. \end{aligned}$$

Example: for $p \geqslant 1$, the l_p norm

$$\|x\|_p = \Big(\sum_i |x_i|^p\Big)^{1/p}.$$

Associated metric $d_{l_p}(x,y) = \|x-y\|_p$. Example:

$$\|x\|_1 = \sum_i |x_i|.$$

Theorem 31

For $V \quad \mathbb{R}^m$, the l_1 distance on V has an exact cut packing of size $\leqslant m(|V| - 1)$.

Proof.

For each dimension i, let P_i be the projection to the ith coordinate. Sort the ith coordinates of the points in V into a sequence $u_{i,1} < \cdots < u_{i,n_i}$ with $n_i \leqslant |V|$. For $1 \leqslant j < n_i$ introduce cuts

$$S_{ij} = \{ v : P_i v \leqslant u_{i,j} \}$$

with weight $y_{ij} = u_{i,(j+1)} - u_{i,j}$.
It is easy to check that this gives an exact cut packing. □

A kind of converse:

Theorem 32

Let y be a cut packing with m nonzero terms in V, defining a metric. Then there is a mapping $\sigma : V \to \mathbb{R}^m$ such that $D^y(u,v) = \|\sigma(u) - \sigma(v)\|_1$.

Isometric embedding.

These two theorems say that a metric can be embedded isometrically into l_1 if and only if it has a cut packing.

If isometric embedding is not available, we may have something weaker. β-distortion l_1-embedding σ:

$$d(u,v)/\beta \leqslant \|\sigma(u) - \sigma(v)\|_1 \leqslant d(u,v).$$

Goal: finding a low-distortion l_1 embedding for an arbitrary metric.

Embedding

For any set $S \quad V$, where $|V| = n$, let $d(u,S) = \min_{v \ S} d(u,v)$. We will try to create an embedding like this, with appropriate sets $S_1, \ldots, S_m$:

$$\sigma_i(u) = d(u,S_i), \qquad i = 1,\ldots,m,$$
$$\sigma(u) = (\sigma_1(u),\ldots,\sigma_m(u)).$$

Clearly $\sigma(u) - \sigma(v)_{\ 1} \leqslant m \cdot d(u,v)$, therefore $\sigma(u)/m$ is an embedding. We need to choose the sets S_i to guarantee low distortion. Since the structure of the metric may be complex, try choosing them randomly. Since we do not know what size random sets are best, choose random sets of different sizes: S_i will have approximate size $n/2^i$.
Let $l = \log n$ (for simplicity, assume $n = 2^l$). For S_i, $i = 2,3,\ldots,l+1$, choose each vertex with probability 2^{-i}.

Expected per-edge distortion

Now $\sigma(u)$ became a random point in $\mathbb{R}^m$. The following lemma gives a constant lower bound First we will lowerbound the expected value $\mathbf{E}(\ \sigma(u) - \sigma(v)\ )_1$ for any pair u, v. Let $c = (1 - e^{-1/4})/2$. The following lemma shows that the distortion in this average is constant.

Lemma 33

We have $\mathbf{E}(\ \sigma(u) - \sigma(v)\ )_1) \geqslant c \cdot d(u,v)/2$.

The proof follows (essentially) by summation from the lemma below. Let $B(u,r)$ and $B^o(u,r)$ be the closed and open ball of radius r around point u. Let ρ_t be the smallest r such that $|B(u,r)| \geqslant 2^t$ and $|B(v,r)| \geqslant 2^t$.

Lemma 34

Suppose that $\rho_t < d(u,v)/2$. *Then* $\mathbf{E}(|\sigma_{t+1}(u) - \sigma_{t+1}(v)|) \geqslant c(\rho_t - \rho_{t-1})$.

Proof.

Let $A = B^o(u, \rho_t)$, $B = B(v, \rho_{t-1})$. We have $|B| \geqslant 2^{t-1}$. Assume without loss of generality $|A| < 2^t$.
We will lowerbound the probability of the event E that $S_{t+1} \cap A = \emptyset$ and $S_{t+1} \cap B = \emptyset$.

$$\mathbf{P}[S_{t+1} \cap A = \emptyset] = (1 - 2^{-(t+1)})^{|A|} \geqslant (1 - 2^{-(t+1)}|A|) > 1/2,$$
$$\mathbf{P}[S_{t+1} \cap B = \emptyset] = (1 - 2^{-(t+1)})^{|B|} \leqslant \exp(-2^{-(t+1)}|B|) \leqslant e^{-1/4}.$$

It follows by independence that $\mathbf{P}(E) \geqslant (1/2)(1 - e^{-1/4}) = c$.
Event E implies $\sigma_{t+1}(u) > \rho_t$ and $\sigma_{t+1}(v) \leqslant \rho_{t-1}$, giving $\sigma_{t+1}(u) - \sigma_{t+1}(v) \geqslant \rho_t - \rho_{t-1}$. Hence

$$\mathbf{E}(\sigma_{t+1}(u) - \sigma_{t+1}(v)) \geqslant \mathbf{P}(E)(\rho_t - \rho_{t-1}) \geqslant c(\rho_t - \rho_{t-1}).$$

From average to probability

We will apply a standard method to get from a lower bound $c \cdot d(u,v)/2$ on the average $\mathbf{E}(\ \sigma(u) - \sigma(b)\ _1)$ to a lower bound with high probability.
A theorem from probability theory:

Theorem 35 (Hoeffding)

Let $X_1, \ldots, X_N$ be independent, equally distributed random variables with average μ and bound $0 \leqslant X_i \leqslant M$. Then for $0 < t$ we have

$$\mathbf{P}[(X_1 + \cdots + X_N)/N - \mu > t] \leqslant e^{-2N(t/M)^2}.$$

Application: we pick the sets S_i^j, $i = 2, \ldots, l+1$, $j = 1, \ldots, N$. Form

$$\overline{\sigma}(u) = (d(u, S_i^j) : i = 2, \ldots, l+1,\ j = 1, \ldots, N).$$

Now $\|\overline{\sigma}(u) - \overline{\sigma}(v)\|_1$ is the sum of N independent random variables bounded by $M = l \cdot d(u,v)$, with expected value $\geqslant \mu = c \cdot d(u,v)/2$. Hoeffding's theorem says for $t = \mu/2$,

$$\mathbf{P}[\ \|\overline{\sigma}(u) - \overline{\sigma}(v)\|_1 < N(\mu/2)] < e^{-2N(\mu/2)^2/M^2} = e^{-Nc^2/(8l^2)}.$$

Choose N here such ($O(l^2 \log n)$) that the right-hand side becomes smaller than $1/n^2$. Then with probability $> 1/2$, we have for all u, v

$$N \cdot l \cdot d(u,v) \geqslant \|\overline{\sigma}(u) - \overline{\sigma}(v)\|_1 \geqslant N \cdot c \cdot d(u,v)/4,$$

where both N and l are $O(\log n)$. Dividing by $N \cdot l$ we obtain an embedding (in an $O(\log^4 n)$ dimensional space) with distortion $O(l) = O(\log n)$.

Improvement

Recall that we apply the metric distortion estimate to

$$\frac{\sum_e c_e d_e}{\sum_{e \quad H} \mathrm{dem}(e) d_e}$$

where H is the demand graph, containing only the source-target pairs. It follows that only the distortion of edges in this graph needs to be estimated. Adapting the above embedding to this, we can reduce the distortion to $\log k$, and embed into a $\log^4 k$-dimensional space, instead of $\log n$.

Applications

- The edge expansion $\delta(S)/|S|$ of a set of a graph. Minimum edge expansion of a graph: via the uniform multicommodity flow problem.
- The conductance of a reversible Markov chain:

$$\min_{S \quad X, 0<\pi(S)\leqslant 1/2} \frac{w(S,\overline{S})}{\pi(S)}.$$

  Demands $\pi(x)\pi(y)$.
- Minimum b-balanced cut, for $0 < b \leqslant 1/2$.

Balanced cut

Finding a $(1/3)$-balanced cut that is within $O(\log n)$ of a minimum bisection cut. This is only pseudo-approximation, since our solution is approximate according two different criteria.

Algorithm 1

$U := \emptyset$, $V := V$. Until $|U| \geqslant n/3$, let W be a minimum expansion set of $G = G_V$, and set $U := U \quad W$, $V := V \setminus W$.

Analysis: let T be a minimum bisection cut. At any iteration, we compare $c_G(W)$ with that of $c(T \cap V)$:

$$c_G(W) \leqslant |W| \cdot C \cdot (\log n) \cdot \frac{c(T)}{n/6}.$$

Summation gives the $O(\log n)$ bound.

Counting problems: the class # P

Definition 36

Function f is in $\#P$ if there is a polynomial-time (verifier) predicate $V(x,y)$ and polynomial $p(n)$ such that for all x we have

$$f(x) = |\{\, y : |y| \leqslant p(|x|) \quad V(x,y) \,\}|.$$

Reduction among $\#P$ problems. The $\#P$-complete problems are all obviously NP-hard.

Repeated tests

How to aproximate a $\#P$ function?
Repeated independent tests will work only if the probability of success is not tiny. More formally, if it is not tiny compared to the standard deviation. Look at Chebysev's inequality, say. Let $X_1, \ldots, X_N$ be i.i.d. random variables with variance σ^2 and expected value μ. Then the inequality says

$$\mathbf{P}[|\sum_i X_i/N - \mu| > t\sigma] \leqslant t^{-2}/N.$$

Suppose we want to estimate μ within a factor of 2, so let $t\sigma = \mu/2$, then $t = \mu/(2\sigma)$,

$$\mathbf{P}[|\sum_i X_i/N - \mu| > \mu/2] \leqslant (1/N)(2\sigma/\mu)^2.$$

This will converge slowly if σ/μ is large.

Example 37

$X_i = 1$ with probability p and 0 otherwise. Then $\sigma^2 = p(1-p)$, our bound is $4(1-p)/(pN)$, so we need $N > 1/p$ if p is small.

DNF satisfaction

Suppose we want to find the number of satisfying assignments of a disjunctive normal form

$$C_1 \quad \cdots \quad C_m.$$

More generally, suppose we need to estimate $|S|$ where

$$S = S_1 \quad \cdots \quad S_m.$$

Suppose that

- We can generate uniformly the elements of S_i for each i.
- We know (can compute in polynomial time) $|S_i|$.
- For each element x, we know

$$c(x) = |\{\, i : x \quad S_i \,\}|.$$

Then we know $M = \sum_i |S_i|$, but we want to know $|S|$.

Pick $I \quad \{1,\dots,m\}$ such that $\mathbf{P}[I=i] = |S_i|/M$. Pick an element $X \quad S_I$ uniformly. Then for each x we have

$$\mathbf{P}[X=x] = \sum_{S_i \quad x} \mathbf{P}[I=i]\mathbf{P}[X=x|I=i] = \sum_{S_i \quad x} \frac{|S_i|}{M}\frac{1}{|S_i|} = c(x)/M.$$

Let $Y = M/c(X)$, then

$$E(Y) = \sum_{x \quad S} \frac{M}{c(x)}\mathbf{P}[X=x] = |S|.$$

On the other hand, $0 \leqslant Y \leqslant M$, so $\sigma \leqslant M \leqslant m|S|$, therefore $\sigma/\mu \leqslant m$, so sampling will converge fast.
We found a FPRAS (fully polynomial randomized approximation scheme) for counting the CNF solutions.

Network reliability

Computing probabilities is similar to counting.
Given graph $G = (V, E)$, each edge is broken with probability p. Let $\mathrm{FAIL}_G(p)$ be the probability that G gets disconnected. Our goal is to estimate this, when it is small (and hence cannot be sampled), say smaller than n^{-4}.
Let c be the size of the minimum cut.

Definition 38

An α-min cut is one with size $\leqslant \alpha c$.
Let E be the event that the graph gets disconnected, and let $E_{>\alpha}$, $E_{\leqslant\alpha}$, $E_{=\alpha}$ be the events that it gets disconnected by a cut of size $> \alpha c \leqslant \alpha c$ or αc respectively.

Then $\mathbf{P}(E) = \mathbf{P}(E_{\leqslant\alpha}) + \mathbf{P}(E_{>\alpha})$. Plan:

- Find an appropriate constant α for which $\mathbf{P}(E_{>\alpha})$ is negligible.
- Represent $E_{\leqslant\alpha}$ by a not too large DNF.

For both steps of the plan, first we estimate the number of minimum cuts.
We will use a random contraction process: in each step, we contract an edge. The probability for minimum cut C to disappear in the step when there are m points left (and hence $\geqslant cm/2$ edges) is $\leqslant 2/m$. So the probability to survive is at least

$$\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\cdots\left(1-\frac{2}{3}\right)=\frac{(n-2)(n-3)\cdots 1}{n(n-1)\cdots 3}=\frac{2}{n(n-1)}.$$

This shows that there are at most $n^2/2$ minimum cuts. A similar argument shows:

Lemma 39

For all $\alpha \geqslant 1$, the number of α-min cuts is at most $n^{2\alpha}$.

The probability of any min cut is $p^c \leqslant \mathrm{FAIL}_G(p) \leqslant n^{-4}$, so we can set $p^c = n^{-(2+\delta)}$ for a $\delta \geqslant 2$.
Now

$$\mathbf{P}[E_{=\alpha}] \leqslant n^{2\alpha} p^{\alpha c} = n^{2\alpha} n^{-\alpha(2+\delta)} = n^{-\alpha\delta}.$$

Using similar estimates repeatedly in a summation one can show:

Lemma 40

We have $\mathbf{P}(E_{>\alpha}) \leqslant n^{-\alpha\delta}(1 + 2/\delta)$.

This allows to choose α such that $\mathbf{P}(E_{>\alpha})$ becomes negligible.

Lemma 41

The $\leqslant n^{2\alpha}$ *cuts of size* $\leqslant \alpha c$ *can be enumerated in polynomial time.*

See the exercises.
Now we can represent $E_{\leqslant\alpha}$ using a DNF.