Analysis of Algorithms

V. Adamchik | CSCI 570 | Fall 2016

Lecture 14 | University of Southern California

Approximation Algorithms and Linear Programming

Based on Chapter 11

Algorithm Design by Kleinberg & Tardos

Computational hardness

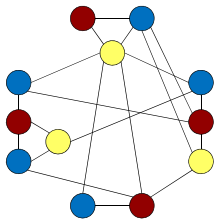
Suppose we are given an NP-hard problem to solve.

Can we develop polynomial-time algorithms that always produce a "good enough" solution?

An algorithm that returns near-optimal solutions is called an *approximation algorithm*.

Graph Coloring

Given a graph G=(V,E), find the minimum number of colors required to color vertices, so no two adjacent vertices have the same color.



Greedy Approximation

Given G=(V,E) with n vertices.

Use the integers {1,2,3, …, n} to represent colors.

Order vertices by degree in descending order.

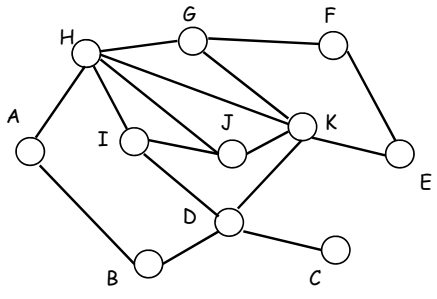
Color the first vertex with color 1.

Go down the list and color every vertex not adjacent to it with color 1.

Remove all colored vertices from the list.

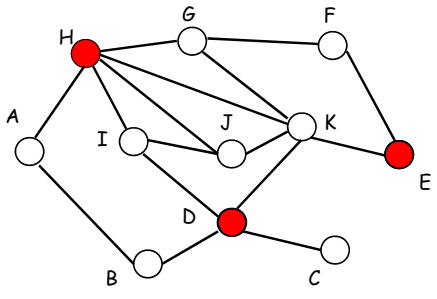
Repeat for uncolored vertices with color 2.

Example



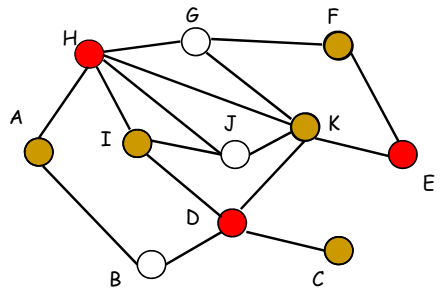
Order: H, K, D, G, I, J, A, B, E, F, C

Example



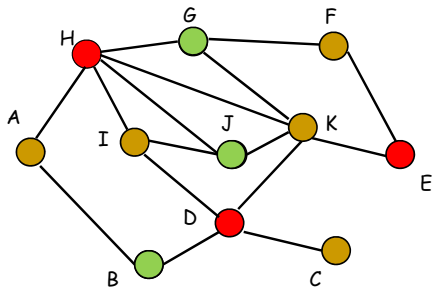
Order: H, K, D, G, I, J, A, B, E, F, C

Example



Order: K, G, I, J, A, B, F, C

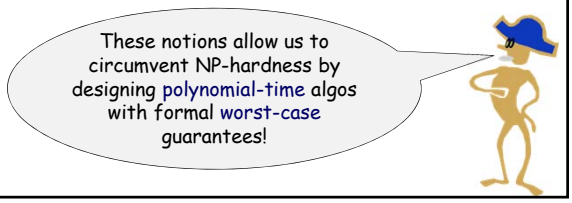
Example



Order: G, J, B

Formal Definition

Let P be a minimization problem, and I be an instance of P. Let ALG(I) be a solution returned by an algorithm, and let OPT(I) be an optimal solution. Then ALG(I) is said to be a *c-approximation* algorithm for some $c > 1$, if for $ALG(I) \leq c \cdot OPT(I)$.



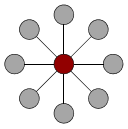
MAX Definition

Let P be a maximization problem, and I be an instance of P. Let ALG(I) be a solution returned by an algorithm, and let OPT(I) be an optimal solution.

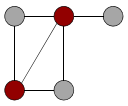
Then ALG(I) is said to be a *c-approximation* algorithm for some $c < 1$, if for $ALG(I) \geq c \cdot OPT(I)$.

Vertex cover

Given G=(V,E), find the smallest $S \subseteq V$ s.t. every edge is incident on a vertex in S.



Next, we design a greedy algorithm for vertex cover...

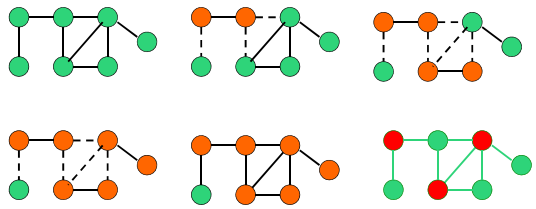


Approximation Algorithm

APPROX-VC(G)

```
C ← Ø
E' ← E[G]
while E' ≠ Ø
    do let (u, v) be an arbitrary edge of E'
     C ← C U {u, v}
     remove every edge in E' incident on u or v
return C
```

Demo

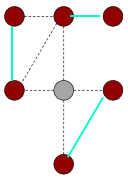


Near optimal, size=6

Optimal, size=3

Vertex cover

Lemma. Let M be a matching (a set of edges) in G, and S be a vertex cover, then $|S| \geq |M|$.



Proof.
S must cover at least one vertex for each edge in M.

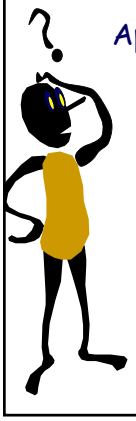
Approximation Vertex Cover

Approx-VC(G):
M ⟵ maximal matching on G
S ⟵ take both endpoints of edges in M
Return S

Theorem. Let OPT(G) be the size of the optimal vertex cover and S = Approx-VC(G). Then $|S| \leq 2 \cdot OPT(G)$

Proof. $|S| = 2\,|M| \leq 2 \cdot OPT(G)$

Approximation Vertex Cover



Theorem. Let OPT(G) be the size of the optimal vertex cover and S = Approx-VC(G). Then $|S| \leq 2 \cdot OPT(G)$

Can we do better than 2 ??

Approximation Vertex Cover

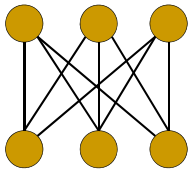
We will show that 2 is a tight bound for this algorithm.

Consider a complete bipartite graph $K_{n,n}$

What is the size of the optimal solution $OPT(K_{n,n})$? n

What is the size of any maximal matching $M(K_{n,n})$? n

Approx-VC($K_{n,n}$) algorithm will return = 2n



Traveling Salesman Problem

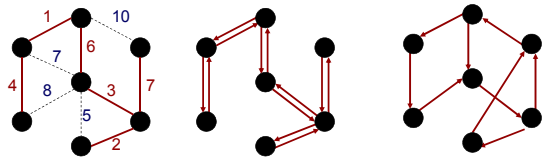
Given an undirected graph G=(V,E) with edge cost $c: E \rightarrow R^+$, find a min cost Hamiltonian cycle.

We will consider a metric TSP.
In the metric TSP, edge costs have to satisfy the triangle inequality, i.e. for any three vertices x, y, z, $c(x,z) \leq c(x,y) + c(y,z)$.

The metric TSP is still NP-hard.

Approximation Algorithm

Approx-TSP(G):
1) Find a MST of G
2) Complete an Euler tour by doubling edges
3) Remove multiply visited edges (shortcuts)



Approximation Metric-TSP

Theorem. Approx-TSP is a 2-approximation algorithm for a metric TSP.

Proof.

$|\text{Approx-TSP}| \le |\text{Euler Tour}| = 2 \cdot |\text{MST}| \le 2 \cdot |\text{OPT}|$

shortcutting decreases the cost.

doubling edges

we can get a spanning tree from HC by removing edges

Christofides Algorithm

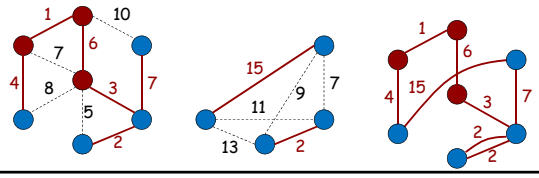
Observe that a factor 2 in the approximation ratio is due to doubling edges; we did this in order to obtain an Eulerian tour.

But any graph with even degrees vertices has an Eulerian tour.

Thus we have to add edges only between odd degree vertices

Christofides Algorithm

Approx-C(G):
T ⟵ MST of G
S ⟵ vertices of odd degree in T
M ⟵ min-cost matching on S
Return: Euler Tour T ∪ M



Christofides Algorithm

Theorem.
Christofides is 3/2 approximation for Metric TSP

The algorithm has been known for over 30 years and yet no improvements have been made since its discovery.

General TSP

Theorem: If P≠NP, then for ∀c>1 there is NO a poly-time c-approximation of general TSP.

Proof. To show Ham-cycle $\le_p$ c-approx TSP.

Start with G and create a new complete graph G' with the cost function

c(u,v) = 1, if (u,v)∈E
c(u,v) = c · n, otherwise (where n = |V|)

Traveling Salesman Problem

Theorem: If P≠NP, then for ∀c>1 there is NO a poly-time c-approximation of general TSP.

Proof.

If G has HC , then |TSP(G')| = n.

If G has no HC , then

$$|TSP(G')| \geq (n-1) + c \cdot n \geq c \cdot n$$

Since the |TSP| differs by a factor c, our approx. algorithm can be able to distinguish between two cases, thus decide if G has a ham-cycle.

Set Covering Problem

Given a collection of subsets

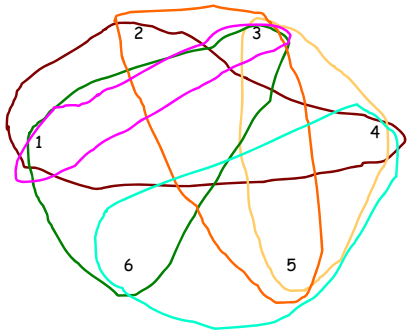
$\mathcal{U} = \{S_1, \ldots, S_m\}$

Find a min-size subset $C=\{S_i, \ldots, S_j\}$ such that C covers $\mathcal{U}$.

Famous NP-hard problem

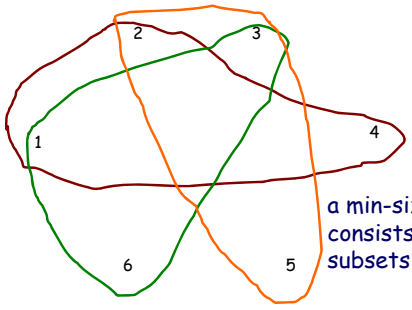
Visualizing Set Cover

S = {1, …, 6}, S_1 = {1,2,4}, S_2 = {3,4,5}, S_3 = {1,3,6}, S_4 = {2,3,5}, S_5 = {4,5,6}, S_6 = {1,3}



Visualizing Set Cover

S = {1, …, 6}, S_1 = {1,2,4}, S_2 = {3,4,5}, S_3 = {1,3,6}, S_4 = {2,3,5}, S_5 = {4,5,6}, S_6 = {1,3}



a min-size cover consists of three subsets

Greedy Algorithm

$\mathcal{U}$ = C [empty cover]
C = {}
While there is uncovered element
    Find the subset S_k covers the most elems
    $\mathcal{U} \leftarrow \mathcal{U} - S_k$
    $C \leftarrow C \cup S_k$
  Return C

How Good of an Approximation?

We'd like to compare the number of subsets returned by the greedy algorithm to the optimal

The optimal is unknown, however, if it consists of k subsets, then any part of the universe can be covered by k subsets!

How Good of an Approximation?

Theorem. If the optimal solution uses k sets, the greedy algorithm finds a solution with at most k ln n sets.

Proof. Since the optimal solution uses k sets, there must some set that covers at least a 1/k fraction of it.

Therefore, after the first iteration n – n/k elements left.

Theorem. If the optimal solution uses k sets, the greedy algorithm finds a solution with at most k ln n sets.

Proof. (contd)

The algorithm chooses the set with most elems which is (n – n/k)/k

Thus, after the second iteration there are n – n/k - (n – n/k)/k elems left

Observe, $n - n/k - (n - n/k)/k = n(1-1/k)^2$

Theorem. If the optimal solution uses k sets, the greedy algorithm finds a solution with at most k ln n sets.

Proof. (contd)

More generally, after y rounds, there are at most

$$n(1-1/k)^y$$

elems left.

Choosing $y = k \ln n$, we get

$$n(1-1/k)^y = n(1-1/k)^{k \ln n} \le n e^{-\ln n} = 1$$

QED.

Linear Programming

Based on Chapter 11

Algorithm Design by Kleinberg & Tardos

Linear Programming

In this lecture we describe linear programming that is used to express a wide variety of different kinds of problems. We can use LP to solve the max-flow problem, solve the min-cost problem, find optimal strategies in games, and many other things.

We will primarily discuss the setting and how to code up various problems as linear programs.

A Production Problem

A company wishes to produce two types of souvenirs: type-A will result in a profit of $1.00, and type-B in a profit of $1.20.

To manufacture a type-A souvenir requires 2 minutes on machine I and 1 minute on machine II.

A type-B souvenir requires 1 minute on machine I and 3 minutes on machine II.

There are 3 hours available on machine I and 5 hours available on machine II.

How many souvenirs of each type should the company make in order to maximize its profit?

A Production Problem

	Type-A	Type-B	Time Available
Profit/Unit	$1.00	$1.20	
Machine I	2 min	1 min	180 min
Machine II	1 min	3 min	300 min

Let $x \geq 0$ be the number of type-A souvenirs and $y \geq 0$ be the number of type-B souvenirs to be made.

The profit: max $(x + 1.2\ y)$

A Production Problem

	Type-A	Type-B	Time Available
Profit/Unit	$1.00	$1.20	
Machine I	2 min	1 min	180 min
Machine II	1 min	3 min	300 min

Let $x \geq 0$ be the number of type-A souvenirs and $y \geq 0$ be the number of type-B souvenirs to be made.

The total amount of time that machine I is used $2x+y$ is and must not exceed 180 minutes.

The total amount of time that machine II is used $x+3y$ is and must not exceed 300 minutes.

A Linear Program

We want to maximize the objective function

$$P = \max (x + 1.2\ y)$$

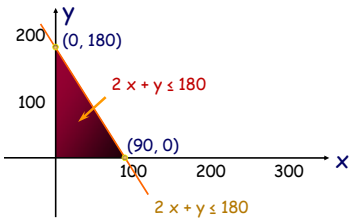
subject to the system of inequalities:

$$2x + y \leq 180$$
$$x + 3y \leq 300$$
$$x \geq 0$$
$$y \geq 0$$

We will solve this problem graphically.

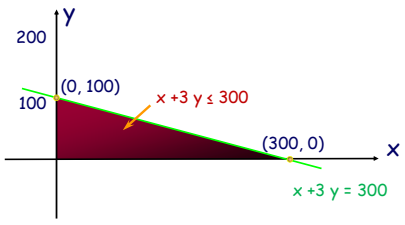
A Production Problem

$$2x + y \leq 180$$



A Production Problem

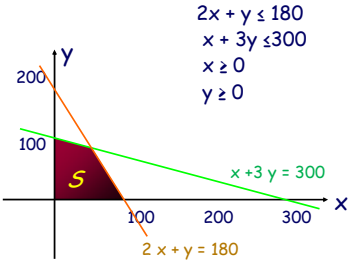
$$x + 3\ y \leq 300$$



A Production Problem

We graph the intersection of four inequalities:

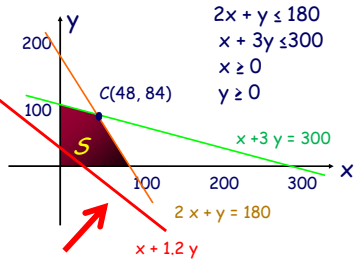
Each point in S is a candidate for the solution of the linear programming problem and is referred to as a feasible solution



A Production Problem

We want to find the feasible point that is farthest in the "objective" direction P=x + 1.2 y

We can see that *P* is maximized at the vertex *C*(48, 84) and has a value of 148.8



Fundamental Theorem

If a linear programming problem has a solution, then it must occur at a vertex, or corner point, of the feasible set *S* associated with the problem.

If the objective function *P* is optimized at *two* adjacent vertices of *S*, then it is optimized at *every point* on the line segment joining these vertices, in which case there are infinitely many solutions to the problem.

Existence of Solution

Suppose we are given a LP problem with a feasible set S and an objective function P.

a) If S is bounded, then LP has a unique optimal solution (P has a max).

b) If *S* is unbounded then the LP solution is unbounded.

c) If S is the empty set, then LP has no solution: infeasible.

The Method of Corners

Graph the feasible set.

Find the coordinates of all corner points (vertices) of the feasible set.

Evaluate the objective function at each corner point.

Find the vertex that renders the objective function a maximum.

If there is only one such vertex, it constitutes a unique solution to the problem.

If there are two such adjacent vertices, there are infinitely many optimal solutions given by the points on the line segment determined by these vertices.

Standard LP form

We say that a maximization linear program with n variables is in standard form if for every variable x_k we have the inequality $x_i \geq 0$ and all other m inequalities. A LP in standard form is written as

$$\max (c_1x_1 + \ldots + c_nx_n)$$

subject to

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

$$\vdots$$

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

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

Standard LP in Matrix Form

The vector c is the column vector $(c_1, \ldots, c_n)$.
The vector x is the column vector $(x_1, \ldots, x_n)$.
The matrix A is the n × m matrix of coefficients of the left-hand sides of the inequalities, and
b= $(b_1, \ldots, b_m)$. is the vector of right-hand sides of the inequalities.

$$\max (c^T x)$$

subject to

$$A x \leq b$$

$$x \geq 0$$

LP in Matrix Form

$\max(x_1 + 1.2\, x_2)$
$2x_1 + x_2 \le 180$
$x_1 + 3x_2 \le 300$
$x_1 \ge 0$
$x_2 \ge 0$

$\Rightarrow$

$\max (c^T x)$
$A\, x \le b$
$x \ge 0$

Here

$x^T = (x_1, x_2)$
$b = (180, 300)$
$c^T = (1, 1.2)$

$A = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$

To get matrix A, rewrite
$2x_1 + x_2 \le 180$
$x_1 + 3x_2 \le 300$

$$\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \le (180, 300)$$

Algorithms for LP

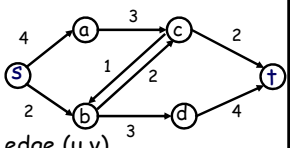
The standard algorithm for solving LPs is the Simplex Algorithm, developed in the 1940s. It's not guaranteed to run in polynomial time.

In 1980 it was shown that LP could be done in polynomial time by something called the Ellipsoid Algorithm (but it tends to be fairly slow in practice).

Later on, a faster polynomial-time algorithm called "interior-point" was developed.

Max-Flow as LP

Consider the example from the network-flow lecture:



Add variable f_{uv} for each edge (u,v).

$0 \le f_{sa} \le 4$, $0 \le f_{ac} \le 3$, $0 \le f_{sb} \le 2$, and so on

Conservation law:

$f_{sa} = f_{ac}$, $f_{sb} + f_{cb} = f_{bc} + f_{bd}$, and so on

What to maximize? $\max(f_{ct} + f_{dt})$

Shortest Path as LP

We are looking for a shortest st-path.

Let d_v denote the shortest path from s to v.

What to maximize? $\min d_t = \max (-d_t)$.

Constrains:

$d_s = 0$, $d_v > 0$, for all vertices $\ne s$

$d_v - d_u \le c(u,v)$ for all edges, $s \notin \{u,v\}$

$d_v \le c(s,v)$ for (s,v) edges.

$-d_u \le c(u,s)$ for (u,s) edges.

Knapsack Problem

maximize $\sum_{k=1}^{n} v_k x_k$

subject to : (1) $\sum_{k=1}^{n} w_k x_k \le W$

(2) $x_k \in \{0,1\}$ for $k = 1..n$.

Interpretation :

$$x_k = \begin{cases} 1 & \text{item k selected} \\ 0 & \text{item k not selected} \end{cases} \quad \text{for } k = 1..n.$$

This is an Integer Linear Programming problem.

Converting to a Standard Form

Write the following LP in standard form.

$\max (3x_1 - x_2)$
$x_1 - 6x_2 + x_3 \le 3$
$7x_2 + x_3 = 5$
$x_1 + x_3 \ge 8$
$x_1 \ge 0$
$x_2 \ge 0$
$x_3 \le 2$

$7x_2 + x_3 \le 5$
$-7x_2 - x_3 \le -5$

$\max (c^T x)$
$A\, x \le b$
$x \ge 0$

Converting to a Standard Form

Write the following LP in standard form.

$\max (3x_1 - x_2)$

$x_1 - 6x_2 + x_3 \le 3$

$7x_2 + x_3 = 5$

$x_1 + x_3 \ge 8$

$x_1 \ge 0$

$x_2 \ge 0$

$x_3 \le 2$

$-x_1 - x_3 \le -8$

$\max (c^T x)$

$A x \le b$

$x \ge 0$

Converting to a Standard Form

Write the following LP in standard form.

$\max (3x_1 - x_2)$

$x_1 - 6x_2 + x_3 \le 3$

$7x_2 + x_3 = 5$

$x_1 + x_3 \ge 8$

$x_1 \ge 0$

$x_2 \ge 0$

$x_3 \le 2$

Add a new slack variable $z_3 \ge 0$, s.t.

$x_3 = 2 - z_3$

$\max (c^T x)$

$A x \le b$

$x \ge 0$

Dual LP: from max to min

Suppose we have a max LP

$$\max (c_1x_1 + \dots + c_nx_n)$$

$$a_{11}x_1 + \dots + a_{1n}x_n \le b_1$$

$$\vdots$$

$$a_{m1}x_1 + \dots + a_{mn}x_n \le b_m$$

Multiply each equation by $y_k \ge 0$ and add them up

$$y_1(a_{11}x_1 + \dots + a_{1n}x_n) \le y_1 b_m$$

$$\vdots$$

$$y_m(a_{m1}x_1 + \dots + a_{mn}x_n) \le y_m b_m$$

Dual LP

$$y_1(a_{11}x_1 + \dots + a_{1n}x_n)$$

$$+\dots$$

$$y_m(a_{m1}x_1 + \dots + a_{mn}x_n) \le y_1 b_m + \dots + y_m b_m$$

Collect terms wrt to x_k.

$$x_1(a_{11}y_1 + \dots + a_{m1}y_n) \quad \ge c_1$$

$$+\dots$$

$$x_n(a_{1n}y_1 + \dots + a_{mn}y_n) \quad \ge c_n$$

$$\le y_1 b_m + \dots + y_m b_m$$

We choose y_k in a way to max $(c_1x_1 + \dots + c_nx_n)$

Dual LP

We choose y_k in a way to max $(c_1x_1 + \dots + c_nx_n)$

$$x_1(a_{11}y_1 + \dots + a_{m1}y_n) \quad \ge c_1$$

$$+\dots$$

$$x_n(a_{1n}y_1 + \dots + a_{mn}y_n) \quad \ge c_n$$

$$\le y_1 b_m + \dots + y_m b_m$$

It follows that if

$$a_{11}y_1 + \dots + a_{m1}y_n \ge c_1$$

then

$$x_1 c_1 \le x_1(a_{11}y_1 + \dots + a_{m1}y_n)$$

Next, compute $x_1c_1 + \dots + x_nc_n$

Dual LP

max

$$x_1c_1 + \dots + x_nc_n \le$$

$$x_1(a_{11}y_1 + \dots + a_{m1}y_n) \quad \ge c_1$$

$$+\dots$$

$$x_n(a_{1n}y_1 + \dots + a_{mn}y_n) \quad \ge c_n$$

min

$$\le y_1 b_m + \dots + y_m b_m$$

We have transformed max $(c_1x_1 + \dots + c_nx_n)$

to min$(b_m y_1 + \dots + b_m y_m)$

So, we end up with a new linear program, in standard minimization form.

Duality

$\max (c^T x)$
$A x \le b$
$x \ge 0$

$\Leftrightarrow$

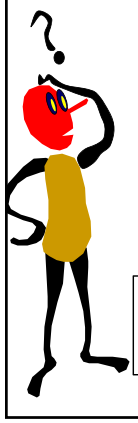
$\min (b^T y)$
$A^T y \ge c$
$y \ge 0$

primal linear program

dual linear program

The optimum of the dual is now an upper bound to the optimum of the primal.

opt(primal) ≤ opt(dual)



Exercise: duality

Consider the LP: $\max(7x_1 - x_2 + 5x_3)$
$x_1 + x_2 + 4x_3 \le 8$
$3x_1 - x_2 + 2x_3 \le 3$
$2x_1 + 5x_2 - x_3 \le -7$
$x_1, x_2, x_3 \ge 0$

Write the dual problem.

$\max (c^T x)$
$A x \le b$
$x \ge 0$

$\min (b^T x)$
$A^T y \ge c$
$y \ge 0$

$\max (c^T x)$
$A x \le b$
$x \ge 0$

Write the dual problem.

$\min (b^T x)$
$A^T y \ge c$
$y \ge 0$

$\max(7x_1 - x_2 + 5x_3)$
$x_1 + x_2 + 4x_3 \le 8$
$3x_1 - x_2 + 2x_3 \le 3$
$2x_1 + 5x_2 - x_3 \le -7$
$x_1, x_2, x_3 \ge 0$

$\min(8y_1 + 3y_2 - 7y_3)$
$y_1, y_2, y_3 \ge 0$

$y_1 + 3y_2 + 2y_3 \ge 7$
$y_1 - y_2 + 5y_3 \ge -1$
$4y_1 + 2y_2 - y_3 \ge 5$

$$A = \begin{pmatrix} 1 & 1 & 4 \\ 3 & -1 & 2 \\ 2 & 5 & -1 \end{pmatrix} \qquad A^T = \begin{pmatrix} 1 & 3 & 2 \\ 1 & -1 & 5 \\ 4 & 2 & -1 \end{pmatrix}$$