EECS 336: Lecture 9: Introduction to Algorithms

Reductions, Decision Problems

Reading: "guide to reductions"

Last Time:

- max flow alg / ford-fulkerson
- duality: max flow = min cut

Today:

- reductions (cont)
- tractability and intractability
- decision problems

Exercise 9.1: Matching to Flow

Setup: Recall Matching to Flow reduction:

Step 1:

i. connect s to each $v \in A$ with capacity 1.

ii. connect t to each $u \in B$ with capacity 1.

iii. add edges $e \in E$ with capacity 1.

Step 2: compute (integral) max flow f

Step 3: matching $M = e \in E : f(e) = 1$

Question: Assuming network flow algorithm runs in $O(m'C')$ where $C' = \sum_{e \text{ out of } s} c(e)$ and $m' = |E|$, what is runtime of bipartite matching algorithm on (A, B, E) with $|A| = |B| = n$ vertices in each part and $|E| = m$ edges?

Reduction Illustrated

Problems	Bipartite Matching	Network Flow
Instance	$x = (A, B, E)$	$y^x = (V^x, E^x, c^x, s^x, t^x)$
Solution	M	f^x

Summary of Reduction

Def: X reduces to Y in polynomial time (notation: $X \leq_P Y$) if any instance of X can be solved in a polynomial number of computational steps and a polynomial number of calls to black-box that solves instances of Y.

Note: to prove correctness of general reduction, must show that correctness (e.g., optimality) of algorithm for Y implies correctness of algorithm for X.

Def: one-call reduction maps instance of X to instance of Y, solution of X to solution of Y. (also called a Karp reduction)

Note: a one-call reduction gives two algorithms:

I. contruction of Y^X instance from X instance.

II. construction of X solution from Y^X solution (with same value.)

Note: the proof of correctness of a one-call reduction gives additional algorithm:

III. construction of Y^X solution from X solution (with same value.)

Note: Only need to consider Y^X instance not general X instance.

Theorem: reduction from "I and II" is correct if I, II, and III are correct.

Proof:

- for instance x of X, let instance of y^x of Y^X be outcome of I.
- II correct $\Rightarrow$ OPT$(x) \geq$ OPT(y^x).
- III correct $\Rightarrow$ OPT$(y^x) \geq$ OPT(x).

$\Rightarrow$ OPT$(x) =$ OPT(y^x).

$\Rightarrow$ output of reduction has value OPT(y).

Exercise 9.2: Perfect Matching

Setup:

- Consider the bipartite graph (A, B, E) with 100 vertices in each part, i.e., $|A| = |B| = 100$.
- Suppose this bipartite graph has a *perfect matching*.
- Consider the reduction to network flow (were the bipartite graph is converted into a network flow instance).

Question:

Can you determine the value of the maximum flow in the network flow instance?

a. Yes, it is 100.
b. No, but it is at most 100.
c. No, and it could be less than or more than 100.

Decision Problems

"problems with yes/no answer"

Def: A decision problem asks "does a feasible solution exist?"

Example: network flow in (V, E, c, s, t) with value at least θ.

Example: perfect matching in a bipartite graph (A, B, E).

Note: objective values for decision problem is 1 for "yes" and 0 for "no".

Note: II and III only need to check "yes" instances.

Note: If solution not needed then reduction is Step I and proof is Steps II and III.

Theorem: perfect matching reduces to network flow decision problem.

Note: Can convert optimization problem to decision problem

Def: the decision problem X_d for optimization problem X has input (x, θ) = "does instance x of X have a feasible solution with value at most (or at least) θ?"

Deciding is as hard as optimizing

Proof: (reduction via binary search)

- given
  - instance x of X
  - black-box $\mathcal{A}$ to solve X_d
- search(A, B) = find optimal value in $[A, B]$.
  - $D = (A + B)/2$
  - run $\mathcal{A}(x, D)$
  - if "yes," search(A, D)
  - if "no," search(D, B)

Reductions for Intractability

"reduce known hard problem Y to problem X to show that X is hard"

Challenge: show problem X is intractable.

"there does not exist algorithm A that solves every $x \in X$ in polynomial time in $|x|$."

Instead: show that solving X enables solving known hard problem Y.

Proof by contradiction:

- assume X is tractable
- can solve Y from X
- so Y is tractable
- contradiction

Tractability and Intractability

Consequences of $Y \leq_p X$:

1. if X can be solved in polynomial time then so can Y.

   Example: X = network-flow; Y = bipartite matching.

2. if Y cannot be solved in polynomial time then neither can X.

Reductions for Intractability

"reduce known hard problem Y to problem X to show that X is hard"

Challenge: show problem X is intractable.

"there does not exist algorithm A that solves every $x \in X$ in polynomial time in $|x|$."

Instead: show that solving X enables solving known hard problem Y.

Proof by contradiction:

- assume X is tractable
- can solve Y from X
- so Y is tractable
- contradiction

Problem Y: 3-SAT

input: blooen formula $f(\mathbf{z}) = \bigwedge_{j=1}^{m} (l_{j1} \vee l_{j2} \vee l_{j3})$

- literal l_{jk} is variable "z_i" or negation "$\bar{z}_i$"
- "and of ors"
- e.g., $f(\mathbf{z}) = (z_1 \vee \bar{z}_2 \vee z_3) \wedge (z_2 \vee \bar{z}_5 \vee z_6) \wedge ...$

output:

- "Yes" if assigment $\mathbf{z}$ with $f(\mathbf{z}) = T$ exists

  e.g., $\mathbf{z} = (T, T, F, T, F, ...)$

- "No" otherwise.

Problem X: INDEP-SET

input: $G = (V, E), k$

ouput: "yes" if $\exists S \subset V$

- satisfying $\forall v \in S, (u, v) \notin E$
- $|S| \geq \theta$

Reduction

Lemma: 3-SAT $\leq_p$ INDEP-SET

Part 1: forward instance construction

convert 3-SAT instance f into INDEP-SET instance (V^f, E^f, θ^f).

- goal: "at least one true literal per clause" $\Leftrightarrow$ "independent set of size at least θ"
- literal $l_{ij} \Rightarrow$ vertices $v_{ij} \in V^f$
- "all clauses true" $\Rightarrow \theta^f = m$
- "literal conflicts" $\Rightarrow$ conflict edges.

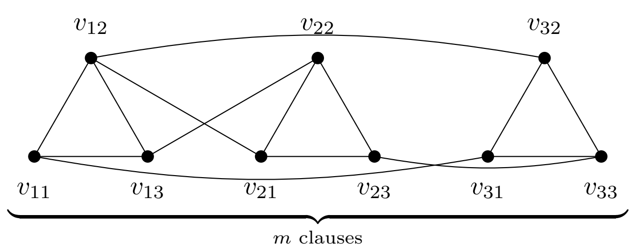
  $\forall i$: $l_{jk} =$ "z_i" and $l_{j'k'} =$ "$\bar{z}_i$" $\Rightarrow (v_{jk}, v_{j'k'}) \in E^f$

- "one representative per clause" $\Rightarrow$ clause edges.

  $\forall j$: $(v_{j1}, v_{j2}),\ (v_{j2}, v_{j3}),\ (v_{j3}, v_{j1}) \in E^f$

Example:

$f(\mathbf{z}) = (z_1 \vee z_2 \vee z_3) \wedge (\bar{z}_2 \vee \bar{z}_3 \vee \bar{z}_4) \wedge (\bar{z}_1 \vee \bar{z}_2 \vee z_4)$



Runtime Analysis: linear time (one vertex per literal.)

Part II: reverse certificate construction

construct assignment $\mathbf{z}$ from S^f

(if (V^f, E^f) has indep. set S^f size $\geq \theta^f = m$ then f is satisfiable.)

1. For each z_r:

   (a) if exists vertex in S^f labeled by "z_r"

   set $z_r = T$

(b) else

set $z_r = F$

Claim: if vertex in S is labeled by "$\bar{z}_r$" then no vertices in S are labeled by "z_r" and z_r is set to False.

(because of conflict edge between vertex labeled "$\bar{z}_r$" and all vertices labeeleed "z_r".)

Claim: S^f independent and $|S^f| \geq m \Rightarrow f(\mathbf{z}) = T$:

- S has $|S| = m$

  $\Rightarrow S$ has one vertex per clause.

- for clause i and $v_{ij} in S$:

  if l_{ij} not negated, then z_i is true (by construction)

  if l_{ij} is negated then z_i is false (by claim)

- So $f(\mathbf{z}) = T$.

Part III: forward certificate construction

construct independent set S^f from $\mathbf{z}$

(if f is satisfiable then (V^f, E^f) has indep set size $\geq m = \theta^f$.)

1. let S' be nodes in (V^f, E^f) corrpesonding to true literals.
2. if more than one vertex in S' in same triangle drop all but one.

   $\Rightarrow S^f$.

Claim: $\mathbf{z}$ satisfies $f(\mathbf{z}) \Rightarrow S^f$ independent and $|S^f| \geq m$

- all clauses have true literal

  $\Rightarrow |S'| \geq m$ and $|S| = m$

- for all $u, v \in S$,
  - u & v not in same triangle.
  - l_u and l_v both true

    $\Rightarrow$ must not conflict

    $\Rightarrow$ no (l_u, l_v) edge in (V^f, E^f).
  - so S^f is independent.