15. Approximate Set Cover and Bin Packing

CPSC 535

Kevin A. Wortman

CALIFORNIA STATE UNIVERSITY FULLERTON

Set Cover: Intuition

- list of **needs**
- list of **services**
  - each service meets some of the needs
- **puzzle:** shortest list of products that meets all the needs?

Set Cover: Formal Definition

set cover problem
input: a universe set X, and family $\mathfrak{F}$ of subsets of X, such that $X = \bigcup_{S \in \mathfrak{F}} S$
output: a minimum size subfamily $\mathfrak{C} \subseteq \mathfrak{F}$ whose members cover all of X, so $X = \bigcup_{S \in \mathfrak{C}} S$

Application: Streaming Services

- **needs:** stream TV shows A, B, C, D, E, F
- $X = \{A, B, C, D, E, F\}$
- **services:** alternative streaming services; each offer only some shows
- $\mathfrak{F} = \{\{A, F\}, \{A, C, E\}, \{B, E\}, \ldots\}$
- **puzzle:** subscribe to smallest number of services that provide all desired shows

Application: Menu Design

- **needs:** menu has a food option available for various dietary needs
- $X = \{carnivore, vegan, kosher, halal, glutenfree, \ldots\}$
- **services:** alternative entrees
- $\mathfrak{F} = \{\{carnivore, halal\}, \{vegan\}, \{kosher, carnivore\}, \ldots\}$
- **puzzle:** design a menu with the fewest number of food options so that everyone can eat something

Set Cover Hardness

- set cover is *NP*-complete
- baseline algorithm: for each subset $\mathfrak{C} \subseteq \mathfrak{F}$, check if the sets in $\mathfrak{C}$ contain all elements, keep track of the smallest such $\mathfrak{C}$
- $\Theta(2^n \cdot n)$ time, slow

Set Cover Approximation Algorithm

function APPROX-SET-COVER($X, \mathfrak{F}$)
    $U_0 = \varnothing$ ▷ still-uncovered elements
    $\mathfrak{C} = \varnothing$
    $i = 0$
    while $U_i \neq \varnothing$ **do**
        // choose set with most currently-uncovered elements
        Find $S \in \mathfrak{F}$ that maximizes $|S \cap U_i|$
        $U_{i+1} = U_i - S$
        $\mathfrak{C} = \mathfrak{C} \cup \{S\}$
        $i = i + 1$
    end while
    return $\mathfrak{C}$
end function

Efficiency Analysis

- while loop: $\Theta(n)$ iterations
  - Find: $\Theta(n)$ time (assuming fast data structure to look up U_i)
  - $U_{i+1} = U_i - S$: $\Theta(n)$ time
- other steps: $\Theta(1)$ time each
- total time $\Theta(n^2)$
- can be sped up to $\Theta(n)$ (CLRS Exercise 35.3-3)

Approximation Ratio

Theorem: APPROX-SET-COVER is a $O(\lg n)$-approximation algorithm

Proof sketch:

- Let $\mathfrak{C}^\star$ be the optimal cover and $k = |\mathfrak{C}$
- $\mathfrak{C}$ covers all of X, and each $U_i \subseteq X$, so $\mathfrak{C}^\star$ covers every U_i
- each U_i can be covered with $\leq k$ sets from $\mathfrak{F}$
- on average, $\mathfrak{C}^\star$ covers n/k elements/set
- so at least one set in $\mathfrak{F}$ covers $\geq n/k$ elements
- APPROX-SET-COVER picks the set that covers the most elements, so each S covers at least n/k additional elements, and

  $$U_{i+1} \leq |U_i| - |U_i|/k = |U_i|(1 - 1/k)$$

Approximation Ratio (continued)

$$U_{i+1} \leq |U_i|(1 - 1/k)$$

- algorithm stops when some $|U_i| = 0$
- as a recurrence,
$$T(n) = (1 - 1/k)n$$
- algebra and log rules show $T(n) \in O(k \lg n)$
- each iteration adds one set to $\mathfrak{C}$, so APPROX-SET-COVER picks $O(k \lg n)$ sets
- k is the optimal number of sets, so
- $\therefore$ APPROX-SET-COVER is a $O(\lg n)$-approximation algorithm $\square$

Set Cover Summary

- set cover is *NP*-complete, exact algorithm takes exponential time
- fast $O(\lg n)$-approximate algorithm
- showed $\Theta(n^2)$ time
- $\Theta(n)$ time is possible

Big Idea: Linear Programming Relaxation

Recall:

- linear programming with real-valued variables is fast (polynomial time)
- integer linear programming (MIP) is *NP*-complete and slow (exponential time)

Idea:

- formulate our problem as a MIP
- "cheat" and solve it as a LP
- round off each solution variable to the nearest integer

Big Idea: Linear Programming Relaxation

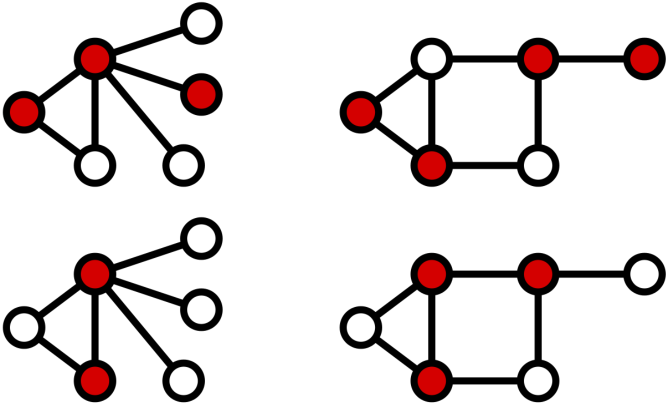
- **LP relaxation:** MIP formulation where no variables are required to be integer
- not correct in general
- **but,** sometimes we can prove an approximate performance ratio
- next: algorithm that uses LP relaxation to solve vertex cover

Review: Vertex Cover

vertex cover problem
input: undirected graph $G = (V, E)$
output: set of vertices $C \subseteq V$, of minimal size $|C|$, such that every edge in E is incident on at least one vertex in C

- *NP*-complete
- previous slides: greedy algorithm, 2-approximate, $\Theta(m + n)$ time
- goal: better performance ratio (smaller)
- expect algo. to be more complicated, slower, or both

Vertex Cover Example



Images credit: Wikipedia user Miym, CC BY-SA 3.0,

https://commons.wikimedia.org/wiki/File:Vertex-cover.svg,

https://commons.wikimedia.org/wiki/File:Minimum-vertex-cover.svg

Review: Formulating Vertex Cover

Variables: for each $v \in V$, create an integer variable x_v such that

$$x_v = 1 \Leftrightarrow v \in C$$

Objective: minimize

$$\sum_{v \in V} x_v$$

Constraints:

$$\begin{array}{lll} 0 \le x_v \le 1 & \forall v \in V & \text{(0 or 1 indicator)} \\ x_u + x_v \ge 1 & \forall (u, v) \in E & \text{(each edge is covered)} \end{array}$$

Review: Vertex Cover Outcomes

- **Infeasible:**
  - never happens
  - $\exists$ a solution: setting all $x_v = 1$ satisfies all constraints
- **Unbounded:**
  - never happens
  - objective is bounded: the objective function is to minimize

$$\sum_{v \in V} x_v;$$

since every $x_v \geq 0$, the minimum objective value is zero, which is finite, so the program is never unbounded

- **Solution:** Construct C as

$$C = \{v \mid v \in V \text{ and } x_v = 1\}$$

Vertex Cover LP Relaxation

Variables: for each $v \in V$, create a <u>real-valued</u> variable x_v such that

$$x_v = 1 \Leftrightarrow v \in C$$

Objective: minimize

$$\sum_{v \in V} x_v$$

Constraints:

$$\begin{array}{lll} 0 \le x_v \le 1 & \forall v \in V & \text{(fuzzy 0 or 1 indicator)} \\ x_u + x_v \ge 1 & \forall (u,v) \in E & \text{(each edge is covered)} \end{array}$$

LP Relaxation Vertex Cover Algorithm

```
function APX-VC-RELAX(G = (V, E))
    C = ∅
    LP = the linear program from the previous slide
    x̄ = SOLVE − LP(LP)                    ▷ assume LP has solution
    for each vertex v ∈ V do
        if x_v ≥ 1/2 then                   ▷ using x_v ∈ x̄
            C = C ∪ {v}
        end if
    end for
    return C
end function
```

Correctness

- LP is never unbounded or infeasible
- must prove that C is a valid vertex cover
- need, for each edge $\{u, v\} \in E$, that $u \in C$ or $v \in C$ (or both)
- LP relaxation has constraints
  $x_u + x_v \geq 1 \quad \forall (u, v) \in E \quad$ (each edge is covered)
- solution $\bar{x}$ satisfies all constraints, so

  $$x_u + x_v \geq 1$$

  is true and

  $$x_u \geq \frac{1}{2} \text{ and } x_v \geq \frac{1}{2}$$
- so the **for** loop adds at least one of u, v to C

Efficiency Analysis

- create LP : $\Theta(n+m)$
- solve LP : polynomial
- post-processing **for** loop: $\Theta(n)$
- total time

$$\Theta(n+m) + \Theta(\text{solve LP}) + \Theta(n) = \Theta(\text{solve LP})$$

- polynomial time

Approximation Ratio

Theorem: APX-VC-RELAX is a 2-approximation algorithm

Proof sketch:

- let $C^\star$ be an optimal vertex cover for G
- need to prove $|C| \leq 2|C^\star|$
- use a "common ground" comparison between $|C|$ and $|C^\star|$
- let

$$\begin{aligned} z^\star &= \text{objective function value of LP} \\ &= \sum_{v \in V} x_v \text{ using each } x_v \in \bar{x} \end{aligned}$$

- we use $z^\star$ to relate $|C|$ to $|C^\star|$

Relating $z^\star$ to $|C^\star|$

- $z^\star$ is objective value $f(C)$ for our relaxed LP
- $C^\star$ is solution to MIP, with more constraints (integer x_v)
- so

$$\begin{aligned} z^\star &\leq f(C^\star) \\ &= |C^\star| \end{aligned}$$

Relating $z^\star$ to $|C|$

- now relate $z^\star$ to $|C|$:

$$\begin{aligned} z^\star &= \sum_{v \in V} x_v \\ &\geq \sum_{v \in V, x_v \geq 1/2} x_v \\ &\geq \sum_{v \in V, x_v \geq 1/2} \left(\frac{1}{2}\right) \\ &= \sum_{v \in C} \frac{1}{2} \\ &= \frac{1}{2}|C| \end{aligned}$$

Completing the Proof of Approximation Ratio

Combine

$$z^\star \leq |C^\star|$$

with

$$z^\star \geq \frac{1}{2}|C|$$

to obtain

$$\frac{1}{2}|C| \leq z^\star \leq |C^\star|$$

or

$$|C| \leq 2 \cdot |C^\star|.$$

Vertex Cover LP Relaxation Summary

- LP relaxation approach:
  - formulate vertex cover as MIP
  - remove integer constraints, solve as LP
  - round each solution variable to nearest integer
- same polynomial runtime as linear programming
- 2-approximation
- compared to greedy algorithm in previous slides, this algo. is
  - simpler
  - slower
- generalizes to **weighted** case (see textbook section 35.5)

Bin Packing: Intuition

- have a collection of **objects**
- want to **pack** them tightly into containers
- **puzzle**: which items go together in each container?

Bin Packing: Formal Definition

bin packing problem
input: a multiset $I = \{i \in \mathbb{Q}, 0 < i \leq 1\}$ of items
output: a partition $I_1, I_2, \ldots, I_k$ of I into k sets, such that the sum of each I_j is at most 1

- bin capacity is 1
- each item $i \in I$ has a fractional size
- ex. $I = \{\frac{2}{3}, \frac{1}{2}, \frac{1}{9}, \frac{1}{2}, \ldots\}$
- k = number of bins used

Bin Packing Examples

Bin Packing Hardness

Bin Packing Approximation Algorithm

Efficiency Analysis

Approximation Ratio

Bin Packing Summary