Algorithmic Perspective on the Vaccine Allocation Problem

CS: 4980 Spring 2020
Computational Epidemiology
Tue, Apr 7
Example: Vaccine Allocation problem

**Input:** Contact network $G = (V, E)$, vaccination budget $B > 0$

**Choice variables:** $x_v \in \{0, 1\}$ for each $v \in V$ ($x_v$ indicates if individual $v$ is to be vaccinated.)

**Possible objective function:** Expected number of individuals infected by an infection (e.g., SIR model) that starts at a random individual and spreads on $G$ with vaccinated individuals removed.

**Constraints:** $\sum_{v \in V} x_v \leq B$ (number of vaccines cannot exceed the budget)
Simplified problem: deterministic infection

An infected node infects all susceptible neighbors in the next time step, after it has become infected.

Implication: if a node in a connected component becomes infected, then all nodes in that connected component will eventually become infected.
Example

• Suppose $B = 1$

• In post-vaccination contact network:
  • If infection source = $a$ then infection size = 4
  • If infection source = $b$ (or $c$ or $d$) then infection size 2
Example

• Suppose $B = 1$

• Expected infection size:

$$\frac{4}{10} (4) + \frac{2}{10} (2) + \frac{2}{10} (2) + \frac{2}{10} (2)$$
Expected Infection Size

- Suppose the original contact network has $n$ nodes and we vaccinate (delete) $B$ of these nodes.

- Suppose this yields $t$ connected components of sizes $c_1, c_2, c_3, \ldots, c_t$.

- Expected size of infection:

$$\frac{c_1}{n-B}(c_1) + \frac{c_2}{n-B}(c_2) + \frac{c_3}{n-B}(c_3) + \ldots + \frac{c_t}{n-B}(c_t)$$
Min Sum-of-Squares Partition (MSSP) problem

**INPUT:** A graph $G = (V, E)$, a positive integer $B$

**OUTPUT:** A subset $S \subseteq V$ of nodes, $|S| = B$, such that if $c_1, c_2, c_3, ..., c_t$ are the sizes of the connected components in $G - S$, then

$$c_1^2 + c_2^2 + c_3^2 + ... + c_t^2$$

is minimum.
Example

• Suppose $B = 1$

• If node in red circle is vaccinated:
  Expected infection size = $4^2 + 2^2 + 2^2 + 2^2 = 28$
• If node in blue box is vaccinated
  Expected infection size = $3^2 + 7^2 = 49$

**Question 1**: can you come up with a 2-sentence argument that with $B = 1$, choosing the node circled red is optimal?
MSSP seeks a balanced partition

Given that

\[ c_1 + c_2 + c_3 + \cdots + c_t = n - B \]

if there were no other constraints on the \( c_i \)'s then

\[ c_1^2 + c_2^2 + c_3^2 + \cdots + c_t^2 \]

is minimized at \( c_i = \frac{n-B}{t} \).
How to efficiently solve this problem?

Degree-based heuristic:
Repeatedly vaccinate node with highest degree in the remaining graph until $B$ nodes are vaccinated

• The performance of the degree-based heuristic can be quite bad.

• $\sim n^2$ (degree-based) vs $\sim \frac{n^2}{2}$ (optimal).
How to efficiently solve this problem?

• **Question 2**: Can you come up with other graphs that are even worse for the degree-based heuristic, making the gap between degree-based and optimal much worse, say 10 times or 100 times even?

• **Question 3**: Other heuristics that seem reasonable to you for solving this problem?
Bad news: MSSP is NP-hard

• **Recall**: This means that if we’re able to come up with an efficient (polynomial-time) algorithm for MSSP, it would imply that many, many other problems (e.g., SAT, TSP, Minimum Vertex Cover, etc.), will all have efficient solutions.

• Since the latter is considered extremely unlikely, the MSSP is extremely unlikely to have an efficient solution.

So what should we do?
Approximation algorithms

For a minimization problem $\Pi$, an algorithm $A$ is an \textit{$\alpha$-approximation algorithm} if:

- $A$ runs in polynomial time
- Cost of solution produced by $A$ is at most $\alpha$ times cost of optimal solution.

An approximation algorithm is a “heuristic” that provides a worst-case guarantee on the gap between its solution and the optimal solution.
Approximation algorithm for MSSP

• **Goal**: To design an efficient $\alpha$-approximation algorithm for MSSP for small $\alpha$.

• Here is an approach from the paper:

Graph Partitioning problems

- Graph Partitioning problems (either via edge removal or node removal) have been studied for decades by the CS community.

- Applications:
  - VLSI design
  - Parallel computing
  - Social network analysis
  - Vaccination allocation

Most graph partitioning problems are NP-hard and are solved by heuristics or by approximation algorithms.
Example: Minimum Cut (MinCut)

**INPUT:** A graph $G = (V, E)$

**OUTPUT:** A partition $(S, V - S)$ (aka “cut”) such that the number of edges between $S$ and $V - S$ is fewest.

- We are looking for a non-trivial solution; so $S \neq \emptyset$ and $V - S \neq \emptyset$.
- This is the “edge version” of the problem because we remove edges to partition the graph.
- An optimal solution in this example has size 2.
Example: Minimum Cut (MinCut) node version

INPUT: A graph $G = (V, E)$

OUTPUT: A partition $(V_1, R, V_2)$ such that there are no edges between $V_1$ and $V_2$ and the size of $R$ is smallest.

Solution needs to be non-trivial, i.e., $V_1 \neq \emptyset$ and $V_2 \neq \emptyset$.

**Question 4:** What is the minimum node cut in this example?
Algorithms for MinCut

• Both the edge version and the node version of MinCut can be solved efficiently (i.e., in polynomial time).

• This is one of the success stories of algorithm design; one way to solve MinCut is by using network flows.
Example: Sparsest Cut (SparseCut)

**Definition:** Given a graph $G = (V, E)$ and a cut $(S, V - S)$, the sparsity of the cut $(S, V - S)$ is

$$
\sigma(S) = \frac{|E(S, V - S)|}{|S| \times |V - S|}
$$

**Numerator:** number of edges that go between $S$ and $V - S$.

**Denominator:** maximum possible edges between $S$ and $V - S$.

$$
\sigma(S_{red}) = \frac{2}{4 \times 4} = \frac{1}{8}
$$

$$
\sigma(S_{pencil}) = \frac{3}{1 \times 7} = \frac{3}{7}
$$
Example: Sparsest Cut (SparseCut)

INPUT: A graph $G = (V, E)$
OUTPUT: A cut $(S, V - S)$ of smallest sparsity $\sigma(S)$.

Question 5: Intuitively, what is the difference between the MinCut and the SparseCut problems?
(Hint: Think about the two problems on a path.)
Example: Sparsest Cut (SparseCut) node version

INPUT: A graph $G = (V, E)$
OUTPUT: A partition $(V_1, R, V_2)$ of such that

$$\frac{|R|}{(|V_1| + \frac{|R|}{2}) \times (|V_2| + \frac{|R|}{2})}$$

is minimized.

**Question 6**: Consider a 5 node path. What is sparsity of the optimal node cut?
Algorithms for SparseCut

• While MinCut has an efficient algorithm, SparseCut is NP-hard.

• But, SparseCut is a relatively old problem and it has a well-known $O(\log n)$-approximation algorithm due to Leighton and Rao (JACM 1999).

Question 7: What does an $O(\log n)$-approximation even mean?
Algorithm for MSSP via a SparseCut algorithm

• A good approximation algorithm for MSSP can be obtained by greedily using a good approximation algorithm for SparseCut.

• A good solution to SparseCut
  • places “few” nodes in $R$
  • and “balances” $|V_1|$ and $|V_2|$  

• So we add $R$ to our set of to-be vaccinated nodes.

• Balancing $|V_1|$ and $|V_2|$ has the effect of minimizing $|V_1|^2 + |V_2|^2$. 
MSSP Algorithm: High-level overview

After the algorithm has proceeded for some iterations, we have:

• a set $B'$ of nodes already set aside for vaccination,
• and connected components $H_1, H_2, \ldots, H_t$ of $G - B'$

Next iteration:

1. Find sparsest cut $R_i$ for each $H_i, i = 1, 2, \ldots, t$.
2. Discard each $R_i$: $|B'| + |R_i|$ is too big, relative to $B$
3. For among the remaining $R_i$'s, add to $B'$ the $R_i$ that is most cost-effective.
4. Replace $H_i$ by the connected components of $H - R_i$
MSSP Result

Theorem: This is an $O((\log n)^2)$-approximation algorithm for the MSSP problem.
Advanced approaches

For the general problem of probabilistic SIR-type models, spectral methods, i.e., methods from linear algebra have been successful.

Thanks for your attention.

Any questions?