Approximating Graph Algorithms Using Metric Embeddings

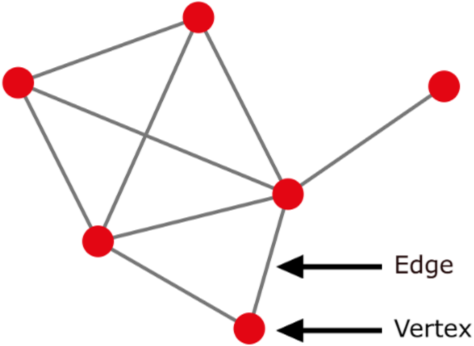
Ethan Luo (mentor: Garrett Tresch)

TAMU

November 19, 2022

What are graphs?

Graphs are a collection of vertices with directed/undirected edges between them.



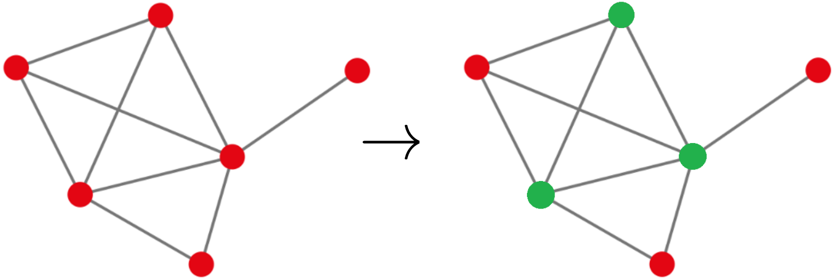
Some simple graph algorithms

Minimum Vertex Cover: using the least number of vertices to "cover" all edges in the graph

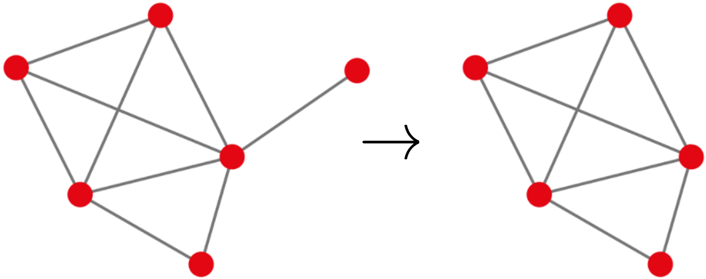
Densest Subgraph: removing some vertices and its corresponding edges from a graph such that the ratio of $\frac{\text{number of edges}}{\text{number of nodes}}$ is maximal

Weighted Maximal Matching: selecting edges from a graph such that the sum of edge weights is maximal and no two edges selected intersect at a vertex

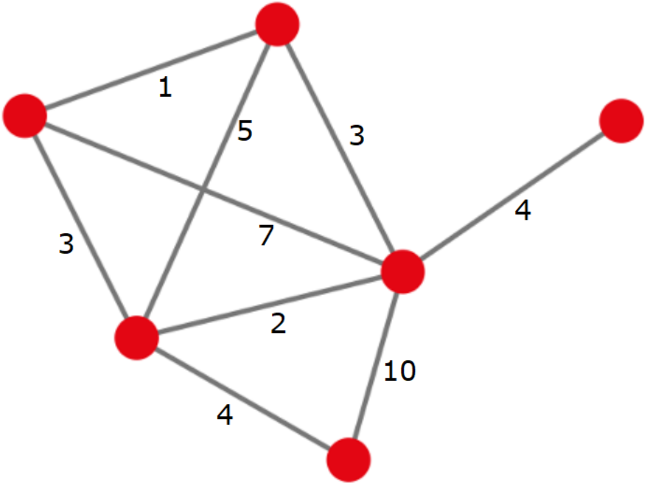
Minimum Vertex Cover



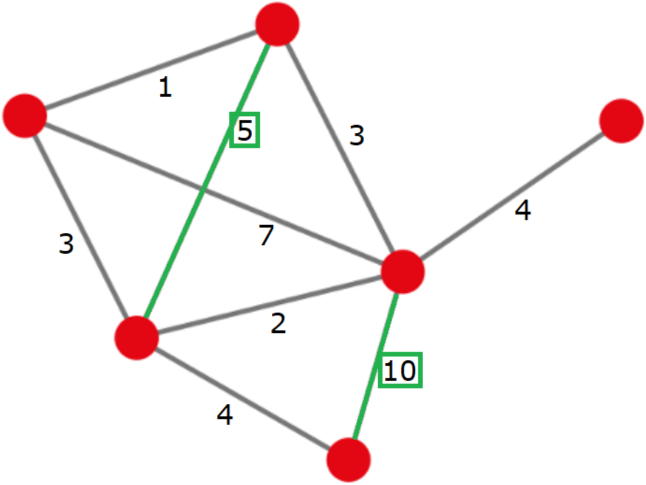
Densest Subgraph



Weighted Maximum Matching



Weighed Maximum Matching



NP-hardness

NP-Hard: a class of problems that can not be solved in polynomial time

All three of the problems, Minimum Vertex Cover, Densest Subgraph, and Weighted Maximum Matching, can be shown to be NP-Hard.

Linear Programming

Linear Programming: maximizing or minimizing some linear objective function with regards to some constraints

A method of solving these is the **Simplex Method**, which checks the vertex points in the convex feasible space of the solution. There are other more modern methods that are utilized more frequently, but the Simplex Method is relatively simple to implement.

Canonical form

Canonical form: has the form, maximize $c \cdot x$ under the constraints $Ax \leq b$

Example: 3 variables 4 inequality constraints.

$$\text{Maximize: } \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\text{Constraints: } \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \\ A_{4,1} & A_{4,2} & A_{4,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Linear Programming Example

Weighted Maximum Matching Problem

Given a graph with n vertices, $\{V_1, ..., V_n\}$, m edges, $\{E_1, ..., E_m\}$, and corresponding edge weights $\{W_1, ..., W_m\}$.

Construct the vector $x = [x_1, ..., x_m]$ of 1's and 0's, where

$$x_i = \left\{ \begin{array}{cr} 1, & \text{if } E_i \text{ is in the graph} \\ 0, & \text{if } E_i \text{ is not in the graph} \end{array} \right\}$$

Our objective will be to maximize $\sum_{i=1}^{m} W_i x_i$.

$$\text{Maximize: } \begin{bmatrix} W_1 \\ ... \\ W_m \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ ... \\ x_m \end{bmatrix}$$

Linear Programming Example

There will be exactly n constraint for this problem, each constraint being on a vertex, specifying that the sum of all edges with endpoints on the vertex is 1 or less.

$$A_{i,j} = \left\{ \begin{array}{lr} 1, & \text{if } E_j \text{ has endpoint } V_i \\ 0, & \text{if } E_j \text{ does not have endpoint } V_i \end{array} \right\}$$

$$\text{Constraints: } \begin{bmatrix} A_{1,1} & \dots & A_{1,m} \\ A_{2,1} & \dots & A_{2,m} \\ \dots & \dots & \dots \\ A_{n,1} & \dots & A_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ \dots \\ x_m \end{bmatrix} \leq \begin{bmatrix} 1_1 \\ 1_2 \\ \dots \\ 1_n \end{bmatrix}$$

Snippet of Linear Program

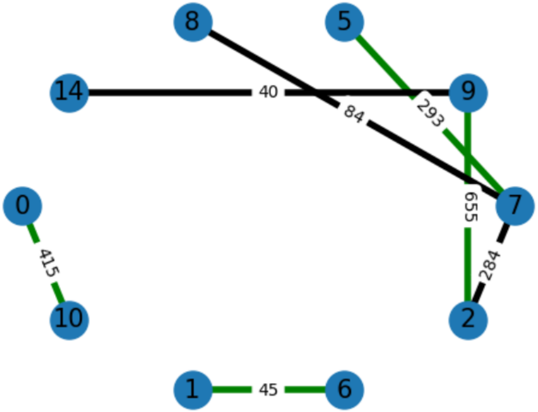
Python Code

```
A = np.zeros((NumberofNodes,edgecount))
for i in range(NumberofNodes):
    for j in range(edgecount):
        if edges[j][0] == i or edges[j][1] == i:
            A[i][j] = 1

b = np.ones(NumberofNodes)
c = np.ones(edgecount)
for i in range(edgecount):
    c[i] = -weight[edges[i]]

x = linprog(c, A_ub=A, b_ub=b, bounds= (0,1))
```

Running the Program



Nodes without edges have been excluded and this is a sample from an online database: sparse.tamu.edu

How else can we relax problems?

Metric Space: a set with a notion of distance, in which the following properties are satisfied

1. $d(x, y) = 0 \Leftrightarrow x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, y) + d(y, z) \geq d(x, z)$

Assuming the set is $\mathbb{R}^n$:
L_p-*space* : $d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$
L_1-*space* : $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
L_2-*space* : $d(x, y) = (\sum_{i=1}^{n} (x_i - y_i)^2)^{\frac{1}{2}}$
L_∞-*space* : $d(x, y) = \max_{i=1}^{n} |x_i - y_i|$

The Sparsest Cut Problem

General form: On an undirected, finite, connected, simple graph G = (V,E), consider a weight function $w_C : E \rightarrow [0,\infty)$ on the edges of G and a non-negative symmetric weight function $w_D : VxV \rightarrow [0,\infty)$ on pairs of vertices. Look to minimize the function

$$\phi_{w_C,w_D}(S) := \frac{\sum_{\{u,v\}\in E: u\in S, v\notin S} w_C(u,v)}{\sum_{u\in S, v\notin S} w_D(u,v)}$$

With unit demand and capacity:

$$\phi_{1,1}(S) := \frac{|E(S, V\backslash S)|}{|S||V\backslash S|}$$

Semi-metric spaces

Semi-metric space: a metric space except d(x,y) $= 0 \not\Rightarrow$ x $=$ y

Since the diagonal is always zero and the matrix is symmetric, we can store all the information of the semi-metric in a vector in $\mathbb{R}^{\binom{n}{2}}$.

$$\text{Distance Matrix: } \begin{bmatrix} A_{1,1} & \dots & A_{1,n} \\ A_{2,1} & \dots & A_{2,n} \\ \dots & \dots & \dots \\ A_{n,n} & \dots & A_{n,n} \end{bmatrix} \text{ Vector: } \begin{bmatrix} A_{1,2} \\ \dots \\ A_{1,n} \\ A_{2,3} \\ \dots \\ A_{2,n} \\ \dots \\ A_{n-1,n} \end{bmatrix}$$

Sparsest Cut Written with Semi-metrics

Note that for any $S \subset V$ we have

$$\frac{\sum_{\{u,v\} \in E : u \in S, v \notin S} w_C(u,v)}{\sum_{u \in S, v \notin S} w_D(u,v)} = \frac{\sum_{\{u,v\} \in E} w_C(u,v) \cdot |1_S(u) - 1_S(v)|}{\sum_{u \neq v \in V} w_D(u,v) \cdot |1_S(u) - 1_S(v)|}$$

Also for any $u, v \in V$ $d_S(u,v) = |1_S(u) - 1_S(v)|$ defines a semi-metric on V. We call all of these semi-metrics generated by sets $S \subset V$, **cut semi-metrics**. From this we see that the sparsest cut problem can be written as

$$\text{SparsestCut}(G, w_C, w_D) = \min_{d_S \text{ cut semi metrics}} \frac{\sum_{\{u,v\} \in E} w_C(u,v) \cdot d_S(u,v)}{\sum_{u \neq v \in V} w_D(u,v) \cdot d_S(u,v)}$$

l_p Semi-metrics

Definition: We say that $d : X \times X \to [0, \infty)$ is an L_p semi-metric if there exists $k \in \mathbb{N}$ and vectors $z_{x:x \in X} \in L_p^k$ such that for all $x, y \in X$ we have that

$$d(x, y) = \|z_x - z_y\|_p$$

Cones

Convex Cone: a collection of vectors with the two properties,

1. $x, y \in C \Rightarrow x + y \in C$
2. $x \in C, \lambda \geq 0 \Rightarrow \lambda x \in C$

Cut semi-metric cone ($\mathbf{CUT_n}$): cone created by all possible cut semi-metrics
L1 semi-metric cone ($\mathbf{NOR_n(1)}$): cone created by all L1 semi-metrics, which coincides with CUT_n
L∞ semi-metric cone ($\mathbf{MET_n}$): cone created by all $L\infty$ semi-metrics, which contains all semi-metrics

Some Results

$$\text{SparsestCut}(G, w_C, w_D) = \min_{d \in \text{CUT}_n} \frac{\sum_{\{u,v\}\in E} w_C(u,v) \cdot d(u,v)}{\sum_{u \neq v \in V} w_D(u,v) \cdot d(u,v)}$$

$$\text{CUT}_n = \text{NOR}_n(1)$$

$$\text{SparsestCut}(G, w_C, w_D) = \min_{d \in \text{NOR}_n(1)} \frac{\sum_{\{u,v\}\in E} w_C(u,v) \cdot d(u,v)}{\sum_{u \neq v \in V} w_D(u,v) \cdot d(u,v)}$$

Note: There is an algorithm for transforming an l_1 semi metric into the positive linear combination of cut semi-metrics (an element of CUT_n).

Overview of Approximation Algorithm

Algorithm 2 Approximation algorithm for Sparsest Cut via LP relaxation

Input: An instance of the Sparsest Cut problem (G, w_C, w_D).
Output: A cut such that $\Phi_{w_C,w_D}(S) \leqslant K \cdot \mathsf{SparsestCut}(G, w_c, w_D)$ where $K = \sup_{d \in \mathsf{LEN}(G)} c_1(V, d)$.

$n \leftarrow |V|$
Solve the LP relaxation of $\mathsf{SparsestCut}$; this yields a semi-metric $d_{\mathsf{LP}} := (d_{\mathsf{LP}}(u,v))_{u,v \in V}$ supported on G.
Find vectors $\{x_v\}_{v \in V} \in \ell_1^k$ witnessing an embedding of d_{LP} with distortion at most K.
Construct the $(n-1)k$ cuts $S_1, \ldots, S_{(n-1)k}$, in the cut cone decomposition for $\{x_v\}_{v \in V} \in \ell_1^k$ using Algorithm 1 as a subroutine.
return S_j that minimizes $\Phi_{w_C,w_D}(S_j)$

Linear Programming for Sparsest Cut Problem

Linear programming relaxation of SparsestCut

Minimize:
$$\sum_{u,v\in V} w_C(u,v)d(u,v)$$

Subject to:
$$\sum_{u,v\in V} w_D(u,v)d(u,v) = 1,$$

$$d(v,v) = 0, \qquad \forall v \in V$$

$$d(u,v) = d(v,u), \qquad \forall u,v \in V$$

$$d(u,v) \leqslant d(u,z) + d(z,v), \qquad \forall u,v,z \in V$$

$$d(u,v) \geqslant 0, \qquad \forall u,v \in V.$$

Preparing to Go Back to L1

The algorithm essentially solves the problem in MET_n, which produces a reduced path semi-metric.

Reduced path semi-metric:

$$d(u, v) = \min\{\sum_{i=1}^{l} \mathrm{len}(x_{i-1}, x_i) : (u = x_0, x_1, ..., x_{n-1}, x_n = v)\}$$

Since we would like to find the actual edges/vertices of the sparsest cut, we have to bring the vector we found in MET_n back into $NOR_n(1)/CUT_n$. This results in distortion.

The **distortion** of a map $f : X \to Y$ is defined as

$$\mathrm{dist}(f) := \sup_{x \neq y \in X} \frac{d_Y(f(x), f(y))}{d_X(x, y)} \sup_{x \neq y \in X} \frac{d_X(x, y)}{d_Y(f(x), f(y))}$$

Defining the Gap

We know $\min\limits_{d \in MET_n} \frac{\sum\limits_{u,v \in V} w_C(u,v)d(u,v)}{\sum\limits_{u,v \in V} w_D(u,v)d(u,v)} \leq \min\limits_{d \in NOR_n(1)} \frac{\sum\limits_{u,v \in V} w_C(u,v)d(u,v)}{\sum\limits_{u,v \in V} w_D(u,v)d(u,v)}$ by union bound. Now let $\mathrm{Gap}_{LP}(G)$ be the largest possible gap between the two functions. We can then prove the following.

Theorem 1 (LP-integrality gap)

Let G = (V,E) be a finite graph, LEN(G) be the set of all reduced path semi-metrics on G, and $c_1(X)$ be the minimum distortion to embed metric space X into L1. Then,

$$\mathrm{Gap}_{LP}(G) = \sup_{d \in LEN(G)} c_1(V, d)$$

A Result from Metric Geometry

How do we get an algorithm and a guarantee for the distortion on the embedding of a reduced path metric into L_1?

Well there is a Theorem of Bourgain in Metric Geometry to help us!

Theorem 2 (Bourgain)

Let $p \in [1, \infty)$ and let X be an n-point semi-metric space, then $c_p(X) = O_p(\log n)$.

Cut Cone Decomposition from L1

Algorithm 1 *Cuts with non-zero coefficients in the cut cone decomposition*

Input: *n vectors $x_1, \ldots, x_n$ in $\mathbb{R}^k$*
Output: *Cuts with non-zero coefficients in the cut cone decomposition for the n vectors.*

for $m = 1$ *to* k **do**
  Find a permutation $\pi_m : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *such that* $x^{(m)}_{\pi_m(1)} \leqslant \ldots \leqslant x^{(m)}_{\pi_m(n)}$
  for $i = 1$ *to* $n - 1$ **do**
    $S^{(m)}_i \leftarrow \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$
  end for
end for
return $S^{(1)}_1, \ldots, S^{(k)}_{n-1}$

We can then try each of the cuts created from this algorithm and use the best.

Conclusions and Extensions

- The general sparsest cut problem, although NP hard can be $O(\log(n))$ approximated in polynomial time by using metric embedding theory.
- Since the sparsest cut problem is often utilized within other graph theory problems and there are many similar problems like Graph Conductance, this argument gives a $O(\log(n))$ approximation for many related problems as well.
- Instead of embedding the problem into an "L_∞" space of semimetrics, we can instead solve the problem in the space of "L_2" semimetrics where we instead solve a semidefinite programming problem (still polynomial time) to obtain a $O(\sqrt{\log(n)} \cdot \log(\log(n)))$ approximation.

Bibliography:

- West, Douglas Brent. Introduction to Graph Theory. Pearson, 2018.
- Lecture notes: METRIC INVARIANTS: ALGORITHMIC AND GEOMETRIC APPLICATIONS (F. Baudier)
- Lecture Notes: ADVANCED GRAPH ALGORITHMS (N. Veldt)
- Leighton, Tom, and Satish Rao. "Multicommodity Max-Flow Min-Cut Theorems and Their Use in Designing Approximation Algorithms." Journal of the ACM, vol. 46, no. 6, 1999, pp. 787–832., https://doi.org/10.1145/331524.331526.

THANK YOU!

Putting It All Together

In the proof of Theorem 1, we get an inequality that proves that Algorithm 2 gives a $K := O(\log(n))$ approximation. If f is a map from the semi-metric obtained from the linear programming problem into L_1 with "Distortion" K then,

$$
\begin{aligned}
\mathsf{SparsestCut}(G, w_C, w_D) &\geqslant \mathsf{SparsestCut}_{\mathsf{LP}}(G, w_C, w_D) = \frac{\sum\limits_{u,v \in V} w_C(u,v)\|f(u)-f(v)\|_1}{K \sum\limits_{u,v \in V} w_D(u,v)\|f(u)-f(v)\|_1} \\
&= \frac{1}{K} \frac{\sum\limits_{j=1}^{(n-1)k} \lambda_{S_j} \sum\limits_{u,v \in V} w_C(u,v) d_{S_j}(u,v)}{\sum\limits_{j=1}^{(n-1)k} \lambda_{S_j} \sum\limits_{u,v \in V} w_D(u,v) d_{S_j}(u,v)} \\
&\geqslant \frac{1}{K} \min_{1 \leqslant j \leqslant (n-1)k} \frac{\sum_{u,v \in V} w_C(u,v) d_{S_j}(u,v)}{\sum\limits_{u,v \in V} w_D(u,v) d_{S_j}(u,v)}.
\end{aligned}
$$