ANOMALY DETECTION ON CONNECTED SUBGRAPHS

XIFAN YU

1. Problem

The anomaly detection problem is concerned with finding a connected subgraph of the input graph-structured data such that average value of data over this subgraph is significantly higher than the average value of the rest of the input. This problem has applications to various real-world problems such as disease outbreak detection and detection of malicious intrusions in a network.

The problem is defined in the following way: The input is a connected undirected graph $G = (V, E)$ and a value function $w : V \to \mathbb{R}^{\geq 0}$. The objective is

$$\max_{\substack{H \subset G: \\ H \text{ connected}}} h(H) = \frac{1}{\sqrt{|V(H)|}} \cdot \sum_{v \in V(H)} w(v)$$

A naive approximation algorithm with $\sqrt[4]{n}$ approximation ratio is to return the maximum objective of the singletons and of the entire graph G:

$$\max\{\max_{v \in V} h(\{v\}), h(G)\}$$

We hope to understand this problem better and attempt to obtain better approximation guarantee. Currently we derived a linear programming relaxation that is efficiently solvable, but no non-trivial integrality gap or rounding algorithm has been found. We also attempted to come up with an SDP relaxation.

2. Linear Programming Relaxation

Since the value functon $w : V \to \mathbb{R}^{\geq 0}$ is non-negative, we may as well consider the squared objective, so that the squared denominator is rid of the square root:

$$\max_{\substack{H \subset G: \\ H \text{ connected}}} h^2(H) = \frac{1}{|V(H)|} \cdot \sum_{\substack{(u,v) \in \\ V(H) \times V(H)}} w(u)w(v)$$

A natural integer program for the objective above would be to associate $x_v \in \{0, 1\}$ for each vertex of G:

$$\frac{1}{\sum_{v \in V} x_v} \cdot \sum_{\substack{(u,v) \in \\ V \times V}} w(u)w(v) \cdot x_u \cdot x_v,$$

and think of $x_v = 1$ if and only if $v \in H$. Now we additional constraints to make sure the subgraph H of the integer program is connected. We introduce variables $y_{u,v} \in \{0, 1\}$ for each unordered pair of vertices $(u, v) \in \binom{V}{2}$, and think of $y_{u,v} = 1$ if and only if $x_u = 1$ and $x_v = 1$. We now use additional cut constraints as follows to make sure the solution of the

Date: Dec 2020.

integer program is a connected subgraph:

$$\begin{aligned}
\max_{x,y} \frac{1}{\sum_{v\in V} x_v} \cdot \left(\sum_{(u,v)\in\binom{V}{2}} w_{u,v}\cdot y_{u,v} + \sum_{v\in V} w_{v,v}\cdot x_v \right) \\
\text{s.t.} \qquad y_{u,v} \le x_u, \qquad & \forall (u,v)\in \binom{V}{2}, \\
y_{u,v} \le x_v, \qquad & \forall (u,v)\in \binom{V}{2}, \\
\sum_{(a,b)\in\delta(S)} y_{a,b} \ge x_v, \qquad & \forall S\subset V : r\in S, v\in \bar{S}, \\
x_v, y_{u,v} \in \{0,1\}, &
\end{aligned}$$

where coefficients $w_{u,v} = 2w(u)w(v)$ and $w_{v,v} = w(v)^2$, and $r \in V$ is some designated root vertex. It is clear that the cut constraints ensure that a vertex v is connected to r in the solution of the integer program whenever $x_v = 1$.

Finally, we scale the variables x, y and get a linear program. We scale the variables x, y so that $\sum_{v\in v} x_v = 1$, to get rid of the denominator in the objective of the program above:

$$\begin{aligned}
(P): \quad \max_{x,y} \left(\sum_{(u,v)\in\binom{V}{2}} w_{u,v}\cdot y_{u,v} + \sum_{v\in V} w_{v,v}\cdot x_v \right) \\
\text{s.t.} \qquad \sum_{v\in V} x_v \le 1, \\
y_{u,v} \le x_u, \qquad & \forall (u,v)\in \binom{V}{2}, \\
y_{u,v} \le x_v, \qquad & \forall (u,v)\in \binom{V}{2}, \\
\sum_{(a,b)\in\delta(S)} y_{a,b} \ge x_v, \qquad & \forall S\subset V : r\in S, v\in \bar{S}, \\
x_v, y_{u,v} \ge 0. &
\end{aligned}$$

Although (P) has exponentially many cut constraints, it has a polynomial time separation oracle, and thus it is efficiently solvable. A violated cut constraint can be found by running $n-1$ maximum flow algorithms from the root vertex r to a sink vertex $v \in V \setminus \{r\}$, with the edge capacity of an edge (a,b) given by $y_{a,b}$. If the maximum flow from r to v is less than x_v, then we discovered a cut that separates r and v with cut value less than x_v, which is a violated cut constraint.

The dual to the linear program (P) is

$$\begin{aligned}
(D): \quad \min_{\alpha,f,g} \ \alpha \\
\text{s.t.} \qquad \sum_{v\in V} f_{u,v} \le \alpha + \sum_{\substack{S: u\in\bar{S} \\ r\in S}} g_{S,u}, \qquad & \forall u\in V, \\
f_{u,v} + f_{v,u} \ge w_{u,v} + \sum_{\substack{S:(u,v)\in\delta(S), \\ u\in\bar{S}, r\in S}} g_{S,u} + \sum_{\substack{S:(u,v)\in\delta(S), \\ v\in\bar{S}, r\in S}} g_{S,v}, \qquad & \forall (u,v)\in E, \\
f_{u,v} + f_{v,u} \ge w_{u,v}, \qquad & \forall (u,v)\notin E, \\
f_{v,v} \ge w_{v,v}, \qquad & \forall v\in V, \\
\alpha, f_{u,v}, g_{S,u} \ge 0. &
\end{aligned}$$

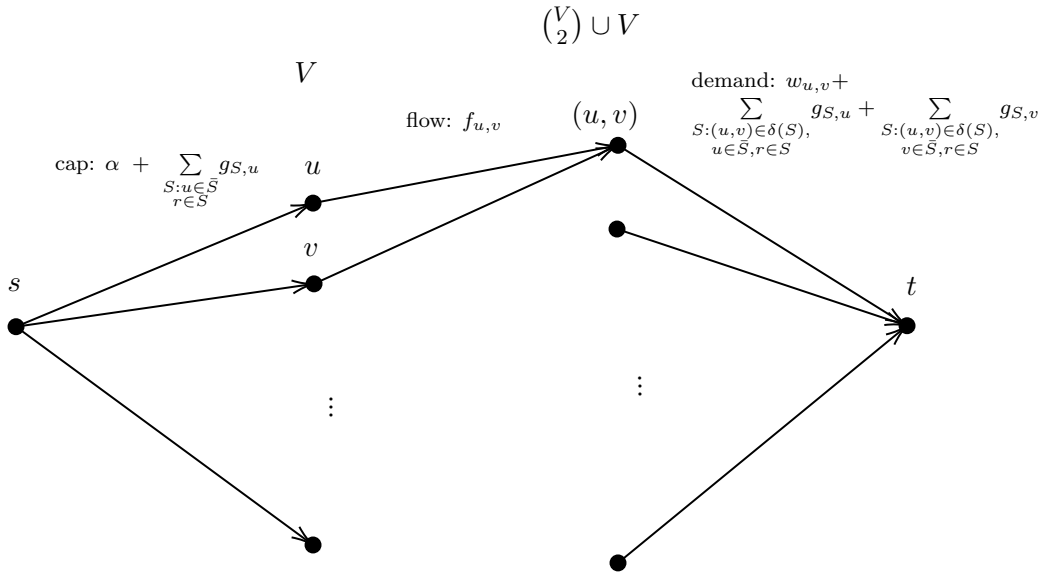

FIGURE 1. Parametric flow corresponding to (D)

Note that without the dual variables $g_{S,v}$ corresponding to the cut constraints in the primal, the dual program (D) can be efficiently solved using parametric flow algorithm described in [2]. Currently, I am trying to understand the dual variables $g_{S,v}$ and how they can be updated using iterative methods to solve the dual efficiently with variants of the parametric flow algorithm. A network flow picture of the dual program (D) is shown in Figure 1.

3. Semidefinite Programming Relaxation

We modified the linear program (P) a bit to get a semidefinite programming relaxation. We use the following vector program, which is equivalent to a semidefinite program.

$$
\begin{aligned}
(SDP): \quad & \text{for parameter } s^*, c^*, \text{ find feasible } x \\
\text{s.t.} \quad & ||x_r||_2^2 = 1, \\
& \sum_{v\in V} ||x_v||_2^2 = s^*, \\
& s^*c^* \le \sum_{v\in V} w_{v,v}||x_v||_2^2 + \sum_{(u,v)\in\binom{V}{2}} w_{u,v}\langle x_u, x_v\rangle, \\
& \langle x_r, x_v\rangle = ||x_v||_2^2, && \forall v \in V, \\
& ||x_u - x_v||_2^2 \le ||x_u - x_w||_2^2 + ||x_v - x_w||_2^2, && \forall u,v,w \in V, \\
& \sum_{(a,b)\in\delta(S)} \langle x_a, x_b\rangle \ge ||x_v||_2^2, && \forall S \subset V : r \in S, v \in \bar{S}.
\end{aligned}
$$

We want $x_v = (1,0,0,...,0)$ if $v \in H$, and $x_v = (0,0,0,...,0)$ otherwise. The parameter s^* indicates the size of the subgraph H, and the parameter c^* corresponds to the objective value.

Currently I am studying the ARV algorithm [1] for sparsest cut and trying to get some idea.

References

[1] Arora, Sanjeev, Satish Rao, and Umesh Vazirani. "Expander flows, geometric embeddings and graph partitioning." Journal of the ACM (JACM) 56, no. 2 (2009): 1-37.

[2] Gallo, Giorgio, Michael D. Grigoriadis, and Robert E. Tarjan. "A fast parametric maximum flow algorithm and applications." SIAM Journal on Computing 18, no. 1 (1989): 30-55.