Decision Support in Production, Logistics and Supply Chain

Arezoo Amiri Christof Brandstetter Tamara Ertl

March 25, 2024

1 Introduction

We only need to add edges with G.add edge(node1, node2, capacity) It automatically adds nodes

Pyvis does not add nodes automatically

net.toggle physics(True), sometimes makes sense to set to False, It makes the graph draggable

Generate C_l cut where $\gamma = 1$ and generate C_m where $\gamma = u$. Then calculate the bisection to obtain a new $\hat{\gamma}$. Then generate a cut with $\hat{\gamma}$ and do this until we find a cut that has less than B large edges.

2 Notation

- A cut is a partition $V = S \cup T$ of the nodes of G such that $s \in S$ and $t \in T$
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$
- Arcs having capacity u are called large arcs
- Arcs having capacity 1 are called small arcs
- $q(C) := \#(\text{large arcs in } C)$
- $p(C) := \#(\text{small arcs in } C)$

3 What's to do?

- Transform the digraph into a NFI graph
  - Create a s-node
  - Create a node for every edge in the digraph
  - Create edges from s to every *edge-node* i_x with capacity 2m

  - Create a node for every node in the digraph
  - Create edges from every *edge-node* to every *node-node* j_1 which is connected by the edges with capacity m
  - Create a node t and connect the *node-nodes* with the t node by edges with capacity m
  - where $m = |E|$

- Implement the bisection algorithm for NFI
  - Generate a minimum cut C_m (minimum Cut to the original graph) and get $q(C_m)$ and $p(C_m)$
  - Generate a least cut C_l (minimum cut of G with the capacity of all arcs set to 1) and get $q(C_l)$ and $p(C_l)$
  - Calculate $\hat{\gamma}$ and generate the cut with capacity-$\hat{\gamma}$-min-cut $\hat{C}$
  - If $cap^{\gamma}(\hat{C}) \leq cap^{\gamma}(C_l)$ and $q(\hat{C}) \notin \{q(C_l), q(C_m)\}$ then generate two new cuts using $((C_l, \hat{C}), (\hat{C}, C_m))$
  - otherwise return $\{C_1, C_2\}$
  - The minimum cut is the smallest set of edges that, when removed, disconnects the graph into two disjoint subgraphs.
  - Identify the arcs $R \in C$ which are removed from the graph
  - C_m is optimal if it contains at least B large arcs with $val(C_m) =$ cap of arcs in cut $-$ cap of removed arcs
  - Assume: Any minimum cut in G contains at most $B - 1$ large arcs
  - C_l denotes a least cut in G, i.e., a cut with least possible number of arcs. Then C_l is optimal if it contains at most B large arcs
  - Assume: Any least cut in G contains at least $B + 1$ large arcs

- Find in the NFI graph a strategy for NFI with budget $B = |E| - \binom{K}{2}$ that has value $K * m$ to get a clique of size K
- Transform it back to find the max-clique
- Use the bisection algorithm for NFI to find large cliques for the benchmark set
  - Suspect, that we have to do this for different K and raise the K's

$R \subseteq E$ is a solution to the u-NFI problem with objective value val(R) which equals the capacity of a minimum s-t-cut in the graph G_R. Minimum s-t-cut of a graph G_R is a minimum cut (cut with least number of arcs, which disjoints s and t) and the capacity of this cut equals the maximum flow of the grpah. (Value of a cut is the sum of the capacities of the arcs in the cut) -¿ cut != removing arcs

So network flow interdiction problem is about finding a subset of arcs R that minimizes the maximum flow from s to t, where all arcs have different capacities (do not need to be different but there are several different ones).

The u-NFI has small (1) and large (u) arcs. Here we need to keep in mind, that the val(C) = val(R_C) = sum of capacity of the B largest arcs

If think the idea of algorithm to find a good solution to the u-NFI is that the we have to compute q-min-cuts (cut with smallest capacity in graph G amongst all cuts with exactly q large arcs) for different values of q. So find the minimum cut with minimal capacity and exactly q large arcs for different values of q and caluclating a q-min-cut is NP-hard.

So we calculate minimum cuts when varying the capacity of the large arcs to obtain cuts with different numbers of q.

If C^γ is a capacity-γ-min-cut for some $\gamma \geq 1$, then it is a q-min-cut for $q = q(C^\gamma)$. This means that if we have a capacity- γ-min-cut C^γ we have found a q-min-cut with q = q(C^γ) or alternatively, by finding a minimum cut for a certain γ we obtain a q-min-cut where q equals the number of large arcs in our minimum cut.

Therefore, we start with C_l and C_m as the q obtained from these are bounds on the optimal q. Then we pick the next γ to check for by doing this bisection. If this cut has a lower capacity than our lower bound (is below our two lines) we do the bisection again for C_l $\hat{C}$ and $\hat{C}$ C_m. An this we do again and again until $\hat{C}$ has a higher capacity.

To finish this up, we will not find a single cut, but rather a set of two cuts which will give us a lower and upper bound on the optimal value.

4 Additional thoughts

1. capacity-γ-min-cut (C^γ) are q-min-cut for $q = q(C^\gamma)$

2. $q(C^{\gamma_1}) \geq q(C^{\gamma_2})$ for $1 \geq \gamma_1 \geq \gamma_2 \geq u \rightarrow$ # of large arcs is decreasing for increasing γ

3. We have bounds on q with $q(C_l)$ and $q(C_m)$

4. Given two cuts C_1 and C_2, the next $\hat{\gamma}$ is chosen by the value, when $cap(C_1) = cap(C_2)$ in the graph $G^{\hat{\gamma}}$

5. If $cap(C^{\hat{\gamma}}) < cap(C_1) = cap(C_2)$ in $G^{\hat{\gamma}}$ bisection is called again for C_1 and $\hat{C}$ and C_2 and $\hat{C}$

6. Otherwise the recursion ends and returns C_1 and C_2

The target here is to find a capacity-γ-min-cut with high γ (equals low # of large arcs) wihle reducing a low cut capacity.