Node Connectivity Augumentation of Highly Connected Graphs

W.Galvez, D.Hyatt-Denesik, A.Jabal Ameli, L.Sanit'a

Presented by Aryan Hemmati

February 17, 2024

Outline

Introduction to the problem and related results

Definition & former results

$\kappa(G)$ is defined as the size of Min-cardinality vertex cut (a set that results in a disconnected graph or a graph with one node). G is said to be d-node-connected iff $\kappa(G) \geq d$.

- Node connectivity of a graph can be computed in polynomial time.
- The approximation problem is APX-hard for $1 \leq k \leq n - n^c$ (with c being any fixed constant)
- The problem can be solved in polynomial time for (weighted, unweighted) $k = n-2, n-3$ and it's proved to be NP-hard for the weighted version of $k = n-4$.
- The cases $k = 1, 2$ yield a 2-approximation factor
- The case $n >> k$ where k is a fixed integer yields a $(4+\epsilon)$-approximation factor. General $\mathcal{O}(1)$-approximation factor is open. Yet a $\mathcal{O}(\log(\min\left\{\frac{n}{n-k}, n-k\right\}))$ is proved that yields $\mathcal{O}(1)$ factor for most cases.

Results and techniques

It is proved in this paper that the $(n-d)$-node-connectivity augmentation problem is NP-hard (& APX-hard) for $d \geq 4, d = \mathcal{O}(n^c)$ for a fixed constant c; which alongside the previous results on the complexity of the problem, completely identifies complexity of the problem in general case. Authors also yield a $\frac{3}{2}$ approximation factor for $d = 4$ in weighted setting, along with a $\frac{4}{3}$ approximation factor in the weighted setting. From now on, we're always given a $(n-d)$-node-connected graph that is not $(n-(d-1))$-node-connected.

- APX-hardness of the case $d = 4$ is proved via a reduction from a bounded variation of SAT. APX-hardness in general case is proved via an approximation-preserving reduction from the case $d = d_0$ to $d = d_0 + 1$ whenever $d_0 = \mathcal{O}(n^c)$.
- Reformulation to d-obstruction covering problem for the complement.
- Computing partial solutions and solving the final simplified instance in polynomial time.

$\frac{3}{2}$ appx. factor for weighted $(n-4)$-NCA

Structure of the problem

Via the reformulation, we're given a graph with no **edge-induced** complete bipartite subgraph $K_{i,j}$ with $i+j>4$, yet includes some $K_{i,j}$'s with $i+j=4$. A subset of the edges are given as links and we're tasked to remove a min-cost set of edges to cover all these obstructions.

- Note that the obstructions consist of $K_{1,3}, K_{2,2}$ which are basically nodes of degree 3, square graphs. Also note that the hypothesis of the problem avoids existence of $K_{1,4}, K_{2,3}$ **edge-induced** subgraphs.

Hence, the main obstructions are the squares since the rest is just an edge cover variant to cover all nodes of degree 3. General relative positioning of squares is always sequential or framed. Hence one can define **lonely nodes, ladders, hexagons**. Finally, the following useful decomposition theorem is given.

Decomposition Theorem

Any 4-obstruction covering instance can be partitioned to **node-induced** subgraphs $G_1, \cdots, G_k, R$ where each G_i is a hexagon or a maximal length ladder and R only consists of lonely nodes.

Maximality is in the sense that if any square shares nodes with the ladder, it should be contained in it. **Lemmas for the proof:**

- Let H, S be hexagon and square subgraphs of G respectively, such that $V(S) \cap V(H) \neq \emptyset$, and $V(S) \not\subseteq V(H)$. Then, in polynomial time, one can find a ladder C in G of length 3, and a labelling of C, such that $V(C) = V(H) \cup V(S)$.
- Let C be a ladder in G with a given labelling. Let S be a square in G, such that $V(S) \cap V(C) \neq \emptyset$, and $V(S) \not\subseteq V(C)$. Then, in polynomial time, one can find a subgraph $D \subseteq G$, such that $V(C) \subsetneq V(D)$, where D is either a ladder with a labelling or a hexagon.

Using the lemmas, one can apply greedy algorithm to finish proof.

The 4-obstruction covering problem can be proved in the special case that G is a ladder or a hexagon, in addition to a set of node covering requirements.
Proof:

- Compute constant size instance to resolve hexagons.
- Avoid particular additional squares via computing constant size instances of ladders of length up to 3.
- Resolve additional square and update node covering requirements. The rest of the problem is solving for simple ladder. For the simple ladder case we simply use a dynamic program to solve the problem efficiently.

Hence one can deduce that the whole "square complex" of the aforementioned decomposition is connected. Hence we define the degree of a "square simplex" (ladders or hexagons). Note that ladders and hexagons have a maximum degree of 4,3.

Here we prove a much harder result about solving the 4 obstruction covering problem on the given decomposition:

4 obstruction covering of bounded degree square complex

If all square simplexes of the decomposition are of degree 1 or 2, then the instance is solved optimally in polynomial time.

In fact, even if some of the simplexes are of degrees higher than 2, one can prove that it is possible to cover all degree 1 or 2 G_i's and all nodes of degree 3 optimally in polynomial time. For proving this result we reduce it to an special case of edge-cover problem that will be defined later on. For now, assume that all edges of the given graph are links and let $c(e) = \infty, \forall e \notin L$.

- For all G_i's of degree 1 or 2, we'll create a pseudo-decision tree of all consequences of covering their corners via the cross-links and define 1-gadgets and 2-gadgets.
- Define the decision graph $(G' = (V', E'), N \subseteq V')$.

It's easily proved (by case-work) that solving this variation of 4-obstruction covering is equivalent to covering nodes of N with minimum cost. We would refer to this variation of edge cover problem as "N-edge cover" problem.

N-edge cover problem

Given a graph $G = (V, E)$ equipped with a cost function $c : V \to \mathbb{R}^{\geq 0}$ and a set of nodes $N \subseteq V$, the N-edge cover problem is solved optimally in polynomial time.

- If $N = V$ then the problem is simply the edge cover problem.
- Otherwise, we add a dummy node v to V and for each node $u \in V/N$, add an edge vu with zero cost. This reduces (obviously in polynomial time) the N-edge cover problem to the edge cover problem, which is known to be solved in polynomial time.

Combining all of the results, we deduce that the 4-obstruction covering of bounded degree square complex is solved efficiently in polynomial time.

Now we compute a partial solution to reduce the general 4 obstruction covering problem to the bounded degree version. Denote the optimal solution of 4-obstruction covering problem by Opt.

- Let N be the set of the corners of all square simplexes of degree 3,4. Compute an N-edge cover in $G = (V, E)$ and denote the optimal solution by $EC_N \subseteq E$. Note that since Opt induces a N-edge cover solution, $c(EC_N) \leq c(Opt)$
- Apply an arbitrary ordering to the edges in EC_N (e.g. timestamps). Then every node in N is satisfied (covered for the first time) by a unique $e \in EC_N$.
- Constructing a satisfaction graph for the set of square complexes of degree 3 or 4, we get a graph $G' = (V' = V \cup D, E')$ in which every g_i has a degree of 3 or 4 (D is a set of dummy nodes).

Now we prove that one could add a polynomial number of dummy nodes to V' and zero cost dummy edges to E' such that G' becomes 4-regular.

Proof:

- For every g_i of degree 3, add a dummy node $g_i{}'$ and a dummy edge $g_i g_i{}'$ with zero cost.
- Now that all g_i is of degree 4, note that all nodes $R = V'/\{g_1, \cdots, g_r\}$ are of degree less than or equal to 4. For all $v \in R$ of degree 1 or 2, add a loop of zero cost. Hence, one can assume that all $v \in R$ are of degree 3 or 4.
- Noting that sum of degrees shall be even, there must have remained an even number of nodes $v \in R$ of degree 3 and one can simply pair them up and connect them together.

Finally, using Petersen's theorem, since the augmented graph is 4-regular, its set of edges can be partitioned to two 2-regular subgraphs. Noting that all dummy edges are of zero cost, cost of all edges in the augmented graph is $c(EC_N) \leq c(Opt)$. Hence, at least one of the subgraphs generated by the Petersen's theorem is of cost at most $\frac{1}{2}c(Opt)$ which induces a covering of G_i's of degree 3 or 4 with the same cost.

Denote the partial solution of the previous section by P_1. To preserve the structure of decomposition, we reduce the instance (using edges of P_1) to the bounded degree instance which (as mentioned) can be solved in polynomial time.

- For each $e \in P_1$, if $e \in \delta(G_i)$ for some $i \in [k]$, let $E = E/\{e\}$.
- Otherwise, $e \in E(G_i)$ for some $i \in [k]$. Therefore, e satisfies (at least) one of the corners of G_i (namely v). Therefore since v is a corner, it is incident to some $uv \in \delta(G_i)$. Then we add a dummy node v' and add an remove uv, add a dummy edge uv' with the same cost. Finally set cost of e to zero.

It's clear that the new instance is a bounded degree instance of 4-obstruction covering. Denote the solution by P_2. Noting that Opt would also induce a solution for the new instance, we deduce that $c(P_2) \leq c(Opt)$. It's now clear that solving this instance of 4-obstruction covering, along with the partial solution P_1 will induce a solution for the original instance with cost bounded by $\frac{1}{2}c(Opt) + c(Opt) = \frac{3}{2}c(Opt)$.

$\frac{4}{3}$ appx. factor for unweighted $(n-4)$-NCA

Preprocessing and iteration steps

Preprocessing step:

- Initially remove all necessary links from E, L as they are contained in all solutions of the problem.
- Note that the bounded degree result still holds in the unweighted case. Compute the partial solution and name it EC_3.

Now we carefully iterate the solution using the following approach to fix it into a total solution of bounded cost. Throughout the solution, we assume that a decomposition into a square complex is given by $G_1, \cdots, G_k, R$.

Cover checker

Given a partial (and potentially infeasible solution) APX, one can find in polynomial time, a subset $I \subseteq [k]$ containing **all** hexagons and ladders of length more than 1 such that APX covers every obstruction of the **node-induced subgraph** $\bigcup_{i \in I} V(G_i)$.

Preprocessing and iteration steps

Proof of the former lemma is pretty straightforward, noting that the decomposition theorem provides us with a labeling of the ladders.

Iteration step:

Throughout the iterative steps, define:

- $APX_i \subseteq L$ is a partial (potentially infeasible) solution
- $H_i \subseteq \{G_1, \cdots, G_k\}$ is the set of square complexes that are completely covered by APX_i
- $Y_i \subseteq APX_i \cap \delta(G[H_i])$ is a subset of edges such that every $e \in Y_i$ has a lonely endpoint

Initially, let $APX_1 = EC_3, H_1 = Y_1 = \emptyset$.

Now we go through an iterative process to make a total solution such that the following holds in every step of iteration.

- Set of obstructions covered by APX_i is a strict superset of the set of obstructions covered by APX_{i-1}.
- $|APX_i \cap (Y_i \cup E[H_i])| \leq \frac{4}{3}|EC_3 \cap (Y_i \cup E[H_i])|$
- $APX_i/(Y_i \cup E[H_i]) = EC_3/(Y_i \cup E[H_i])$

Note that first condition implies $H_{i-1} \subseteq H_i$. Along with the fact that $EC_3 \subseteq APX_i$ covers all nodes of degree 3, this proves that $EC_3 \subsetneq APX_i$. (case-work to prove that there cannot be two squares left uncovered in a square simplex) We will continue the process until $\{G_1, \cdots, G_k\}/H_i$ consists only of ladders of length one and then resolve the remaining instance with another lemma. Otherwise let $G_j \in \{G_1, \cdots, G_k\}/H_i$ be a hexagon or ladder of length at least 2:

- G_j is a hexagon.
- G_j is a ladder of length at least 2 with no hexagon.

Reducing the problem to simple complexes

Let $H_0 \subseteq \{G_1, \cdots, G_k\}$ denote the ladders and hexagons covered by EC_3. We set $APX_0 := EC_3 \cap E[H_0]$, and remove these edges from G to get a new instance of 4-obstruction covering, equipped with a square complex (derived from decomposition theorem) where all square simplexes are of degree 3,4.
We then use result of case 1 repetitively to cover all hexagons and the result of case 2 to break ladders to ladders of length 1 with degree 3,4.

Wrapping up the complexity of (n-4)-NCA

Reducing 4-obstruction covering to a SAT variant

3-SAT-(2,2)

A variation of 3-SAT where each literal and its negation have both appeared twice.

One can now reduce this to 3-SAT-4 which is known to be NP-hard. Now assume that I is an instance of 3-SAT-4:

Let $G_I = (\emptyset, \emptyset)$

- For each clause C, add node C to G_I.
- For each variable $x \in I$, we add a subgraph H_x to G_I, where H_x is defined by variable nodes $x_1, \cdots, x_6$, and edges $\{x_1x_2, x_2x_3, x_3x_4, x_4x_5, x_5x_6, x_1x_6, x_2x_5\}$. The edges $\{x_1x_2, x_2x_3, x_4x_5, x_5x_6\}$ are set to be links.
- For clauses C_1, C_2 containing the positive literals of x, we add edges $\{x_1C_1, x_4C_2\}$ to both G and L. For clauses C_3, C_4 containing the negative literals of x, we add edges $\{x_3C_3, x_6C_4\}$ to both G_I, L.

Verifying the reduction

It can be verified that the constructed graph G_I doesn't have any $K_{1,4}, K_{2,3}$ as node-induced subgraphs and therefore is a valid 4-obstruction covering instance. Furthermore, one can see that if I has k variables, then the 4-obstruction covering instance of G_I has a feasible solution of size $4k$. Conversely, if G_I has a feasible solution of size $4k$ then I is a satisfiable $3 - SAT_(2, 2)$ instance.
This, along that $3 - SAT - (2, 2)$ is NP-hard (and in fact APX-hard) proves that 4-obstruction covering is a NP-hard (and APX-hard) problem.

Finally, we extend this construction to the general case:

Let $G_I = (\emptyset, \emptyset)$

- For each clause C, add node C to G_I.
- For every clause node C, we add $d-4$ dummy nodes to G'_I that are adjacent to C, so that C is degree $d-1$.
- For subgraph H_x, with nodes $x_1, \cdots, x_6$, we add $d-4$ dummy nodes to G'_I that are adjacent to x_3 and $d-4$ dummy nodes adjacent to x_6. We add $d-4$ nodes to G' that are adjacent to both x_1, x_5, which we denote by $y_1, \cdots, y_{d-4}$ and we add $d-4$ nodes that are adjacent to both x_2, x_4, which we denote by $z_1, \cdots, z_{d-4}$. None of these new edges are added to L'_I and we have $L'_I = L_I$.