Short Note on Edge Connectivity Augmentation

Michael Bussieck *

Abstract

Consider a network $N = (V, E_c, c)$ with an integer valued capacity function $c: V \times V \to \mathbb{Z}_+$ and let $k$ be a positive integer. What is the minimal total increase $\gamma$ by which the individual capacities must be increased such that the edge connectivity number is at least $k$?

Clearly the defect $r(u) := \max(k - \lambda(u), 0)$ summing over all vertices, called $\gamma_T$ is a lower bound for $\gamma$. Kajitani and Uno [7] proved that if $(V, E_c)$ is a tree then $\gamma = \gamma_T$. We extend this result to the larger class of acyclic digraphs.

Frank [4] gives a min-max formula for $\gamma$ which is proved using Maders [8] splitting theorem. In order to obtain an efficient implementation of the resulting strong polynomial time algorithm, one must carry out some reduction and splitting operations which, in turn, entail performing several maximal flow computations.

We give an implementation which significantly decreases the time complexity of the reduction phase, and substantially reduces the running time of the entire algorithm. Furthermore we give some computational results.

1 Introduction

Let $\phi: V \times V \to \mathbb{Z}_+$ be a weight function on the possible edges of a directed or undirected graph $G$ with vertex set $V$. We define the cut function $\delta_\phi: 2^V \to \mathbb{Z}_+$ as

$$\delta_\phi(T) := \sum_{ij \in T} \phi(ij) \quad \forall T \in 2^V.$$ 

For $V \setminus T$ we write $\hat{T}$. Let $N = (V, E_c, c)$ be a network with a capacity function $c: V \times V \to \mathbb{Z}_+$, a set of vertices $V$, and a set of (directed) edges $E_c := \{ij | c(ij) > 0\}$. We define the edge connectivity number of $N(V, E_c, c)$ as

$$\lambda_c(V) := \min_{\emptyset \subset T \subset V} \delta_c(T).$$

If it is clear from the context we write $\lambda_c$ instead of $\lambda_c(V)$.

**Augment:** Consider a network $N = (V, E_c, c)$ and a positive integer $k$. Find a capacity function $c'$ such that, in the resulting network $N' = (V, E_{c+c'}, c+c')$ the edge connectivity number $\lambda_{c+c'}$ is at least $k$, under the objective to minimize the sum of increments $\gamma := \sum_{i,j \in V} c'(ij)$.

A subpartition of $V$ is a family $\{X_1, \ldots, X_t\}$ of disjoint, proper subsets of $V$. The following fundamental theorem and its constructive proof are due to Frank [4]

**Theorem 1** Given an instance for problem **Augment**, there is a capacity function $c'$ with total increase $\gamma$ which satisfies $\lambda_{c+c'} \geq k$ if and only if, for every subpartition $\{X_1, \ldots, X_t\}$, we have

$$\tilde{\gamma} := \sum_{i=1}^t r(X_i) \leq \gamma, \quad \text{and} \quad \tilde{\gamma} := \sum_{i=1}^t r(\hat{X}_i) \leq \gamma$$

with $r(T) := \max(k - \delta_c(T), 0)$.

---

*TU Braunschweig, Abteilung für Mathematische Optimierung, Pockelsstraße 14, D-38106 Braunschweig, Germany
2 Augmenting Acyclic Digraphs

In 1986 Kajitani and Ueno [7] solved AUGMENT when \((V, E_c)\) is a directed tree. They observed that the defect of the subpartition in (1) is maximal for the subpartition that consists of all single vertices. This means that we must count the defect at the vertices only, and then we know the total increase \(\gamma\) we must augment. We will extend this result to a larger group of graphs, the acyclic digraphs (a digraph with no directed cycle).

**Lemma 1** For a network \(N = (V, E_c, c)\) with an acyclic digraph \((V, E_c)\) we have

\[
\gamma_T := \sum_{v \in V} r(v) = \gamma \quad \text{and} \quad \tilde{\gamma}_T := \sum_{v \in V} r(\bar{v}) = \tilde{\gamma}
\]

**Proof.** Clearly \(\gamma_T \leq \gamma\) and \(\tilde{\gamma}_T \leq \tilde{\gamma}\) hold, because \(\{v_1, \ldots, v_{|V|}\}\) is a subpartition of \(V\). Now suppose that \(\langle v, u \rangle\) holds. Let \(P := \{X_1, \ldots, X_t\}\) be the subpartition with maximal outgoing defect \(R(P) := \sum_{X_i \in P} r(X_i)\), and among these, with minimal cardinality \(\sum_{X_i \in P} |X_i|\). There must be a set \(X_{i_0}\) with \(|X_{i_0}| > 1\). Consider the following two cases:

1. \(\delta(X_{i_0}) < \delta(u)\) holds for all \(u \in X_{i_0}\): Each vertex \(u \in X_{i_0}\) has a leaving edge \((u, v)\) with \(v \in X_{i_0}\). We follow these edges. After at most \(|X_{i_0}|\) steps we reach a vertex we visited before, contradicting that \((V, E_c)\) is acyclic.

2. There is a \(u \in X_{i_0}\) with \(\delta(X_{i_0}) \geq \delta(u)\): We exchange \(X_{i_0}\) by \(\{u\}\) in \(P\) obtaining \(P'\). Since \(R(P)\) is maximal we have \(\delta(X_{i_0}) = \delta(u)\), hence \(R(P) = R(P')\). Since \(|X_{i_0}| > 1\) we have \(\sum_{X_i \in P} |X_i| > \sum_{X_i \in P'} |X_i|\), which contradicts the minimal cardinality of \(P\).

The equality for the incoming defect follows analogously. \(\square\)

This result together with theorem 1 leads to the following corollary.

**Corollary 2** Given a network \(N = (V, E_c, c)\) with an acyclic digraph \((V, E_c)\) and a positive integer \(k\), there is a capacity function \(c'\) with total increase \(\gamma := \max(\gamma_T, \tilde{\gamma}_T)\) such that \(\lambda_{c+c'} \geq k\).

3 Implementing Frank’s Algorithm

From the constructive proof of theorem 1 (cf. [4]) we obtain the following algorithm for AUGMENT.

Let \(N = (V, E_c, c)\) and \(k\) be an instance for AUGMENT. Add a vertex \(s\) to \(V\) obtaining \(\hat{V}\). Let \(\hat{c}: \hat{V} \times \hat{V} \rightarrow Z_+\) with \(\hat{c}(uv) = c\) and \(\hat{c}(us) = \hat{c}(su) = k\) for all \(u \in V\). Hence

\[
\hat{\delta}_c(T) \geq k \quad \text{hold for all } \emptyset \subset T \subset V. \tag{2}
\]

During the algorithm we change \(\hat{c}\) such that \(\hat{\delta}_c(s) = \hat{\delta}(s) = 0\) then \(\lambda_{\hat{c}c}(V) \geq k\). Using this condition we reduce \(c(us)\) and \(c(su)\) for all \(u \in V\).

**Reduction**: Consider the edges incident with \(s\). Proceeding edge by edge, reduce the capacity of the current edge as much as possible without violating (2). After this REDUCTION we have \(\hat{\delta}_c(s) \leq \gamma\) and \(\hat{\delta}_c(s) \leq \gamma\) (cf. [4]).

**Splitting**: Under the assumption that (2) holds, a theorem of Mader [8] guarantees that we can always find a pair \(\{u, v\}\) with \(\hat{c}(u, s) > 0, \hat{c}(s, v) > 0\) and a positive integer \(\beta\) such that the following splitting operation can be carried out without violating (2).

\[
\hat{c}(u, s) := \hat{c}(u, s) - \beta, \quad \hat{c}(s, v) := \hat{c}(s, v) - \beta, \quad \text{and} \quad \hat{c}(u, v) := \hat{c}(u, v) + \beta
\]

After a finite number of splitting operations we obtain, \(\delta_c(s) = \hat{\delta}_c(s) = 0\). Hence \(\lambda_{\hat{c}c}(V) \geq k\) and \(\sum_{i,j \in V} \hat{c}(i, j) - c(i, j) \leq \gamma\).
The maximal number \( \alpha \) by which we can reduce \( c(u, s) \) (\( c(s, u) \) can be treated analogously) without violating (2) is

\[
x := \min \left\{ \min_{\stackrel{v \in V \setminus T}{s \in \hat{T}}} \delta_\varepsilon(T) - k, k \right\}.
\]

So we have to compute a proper subset \( T^* \) of \( V \) containing \( u \) with \( \delta_\varepsilon(T^*) = \min_{v \in \hat{T}} \delta_\varepsilon(T) \) or decide that \( \min_{v \in \hat{T}} \delta_\varepsilon(T) \geq 2k \). Using the max-flow–min-cut theorem of Ford and Fulkerson \( [3] \) we obtain \( T^* \) and so \( \alpha \) by computing some maximal flows in \( \tilde{N} = (\hat{V}, E_\varepsilon, \tilde{c}) \). Let \( \mathcal{MF}_N(S, T) \) denote the value of a maximal flow from the sources \( S \subset V \) to the targets \( T \subset V \) in the network \( N \). Now a plain procedure for computing \( T^* \) could be,

\[
\delta_\varepsilon(T^*) = \min_{w \in V \setminus \{u\}} \mathcal{MF}_N(u, \{w, s\}).
\]

For each \( u \in V \) we have to perform \( |V| - 1 \) calls to a max-flow routine which effects in \( 2|V|(|V| - 1) \) max-flow calls for the complete reduction.

We improve this number using the next lemma.

**Lemma 2** Let us be not the last edge in the reduction sequence \((r.s.)\), then

\[
\mathcal{MF}_N(u, s) = \delta_\varepsilon(T^*) \quad \text{or} \quad \delta_\varepsilon(T^*) \geq 2k.
\]

**Proof.** From the mf-mc theorem we obtain a set \( \emptyset \subset \hat{T} \subset \hat{V} \) with \( u \in \hat{T}, \ s \notin \hat{T}, \) and \( \delta_\varepsilon(\hat{T}) = \mathcal{MF}_N(u, s) \). If \( \hat{T} \subset V \) then \( \delta_\varepsilon(\hat{T}) = \delta_\varepsilon(T^*) \) and \( \alpha = \min\{\delta_\varepsilon(\hat{T}) - k, k\} \). Now suppose \( \hat{T} \neq V \). Since \( u \) is not the last edge in the r.s. there is another \( v \in V \) with \( \tilde{c}(v, s) = k \). Hence \( \delta_\varepsilon(T^*) \leq \delta_\varepsilon(T') \) and \( \alpha = k \).

Except for the last edge in the r.s. we compute \( \alpha \) calling the max-flow routine once. Only for this last edge we calculate \( \alpha \) using the procedure described in (3). Hence we obtain \( 4(|V| - 1) \) max-flow calls for the complete reduction.

Let \( \{u, v\} \) be a valid pair in the splitting phase. We have

\[
\beta = \min \left\{ \min_{\stackrel{v \in V \setminus \{u, \}}{s \in \hat{T}}} \delta_\varepsilon(T) - k, \min_{\stackrel{v \in V \setminus \{u, \}}{s \in \hat{T}}} \delta_\varepsilon(T) - k, \tilde{c}(u, s), \tilde{c}(s, v) \right\}.
\]

(4)

Similar to \( \alpha \) we can determine \( \beta \) by computing

\[
\delta_\varepsilon(T') = \min_{w \in V \setminus \{u, v\}} \mathcal{MF}_N(\{u, v\}, \{w, s\}) \quad \text{and} \quad \mathcal{MF}_N(\{u, v\}, \{w, s\}) = \min_{w \in V \setminus \{u, v\}} \mathcal{MF}_N(\{w, s\}, \{u, v\}),
\]

and search the minimum among the four numbers in (4). This leads to \( 2p(|V| - 2) \) max-flow calls, where \( p \) denotes the number of pairs considered in the splitting phase.

**Lemma 3** If \( \delta_\varepsilon(V) \geq \min(\tilde{c}(u, s), \tilde{c}(s, v)) + k \) then

\[
\mathcal{MF}_N(\{u, v\}, s) = \delta_\varepsilon(T') \quad \text{or} \quad \delta_\varepsilon(T') \geq \min(\tilde{c}(u, s), \tilde{c}(s, v)).
\]

If \( \delta_\varepsilon(s) \geq \min(\tilde{c}(u, s), \tilde{c}(s, v)) + k \) then

\[
\mathcal{MF}_N(s, \{u, v\}) = \delta_\varepsilon(T'') \quad \text{or} \quad \delta_\varepsilon(T'') \geq \min(\tilde{c}(u, s), \tilde{c}(s, v)).
\]

**Proof.** Consider the first part of the lemma. Let \( \emptyset \subset \hat{T} \subset \hat{V} \) be the set with \( u, v \in \hat{T}, \ s \notin \hat{T}, \) and \( \delta_\varepsilon(\hat{T}) = \mathcal{MF}_N(\{u, v\}, \{s, \}) \). If \( \hat{T} \subset V \) then \( \delta_\varepsilon(\hat{T}) = \delta_\varepsilon(T') \) and \( \alpha = k \). Hence we have \( \delta_\varepsilon(\hat{T}) = \delta_\varepsilon(T'') \) and obtain the second alternative. The second part of lemma 3 follows analogously.
If the conditions of lemma 3 hold for a pair \( \{u, v\} \) we can determine \( \beta \) by calling the max-flow routine twice. Since \( k \) is an upper bound for \( \tilde{c}(u, s), \tilde{c}(s, v) \) we have: if \( p > 2k \) then \( 2(p - 2k) + 4k(|V| - 2) \) max-flows must be computed for the complete splitting. In [4, 1] we find some techniques which bound \( p \) to \( 3|V| \).

Let \( \mu(V) \) denote the complexity of an algorithm for the max-flow problem with an input network having vertex set \( V \). We attain the time complexity \( O(|V| \cdot \mu(V)) \) for the reduction phase and \( O(\min(k, \frac{|V|}{2}) \cdot |V| \cdot \mu(V)) \) for the splitting phase, instead of \( O(|V|^2 \cdot \mu(V)) \) for reduction and splitting in the plain implementation.

## 4 Numerical Investigations

We created random networks using the RMFGEN generator of Goldfarb and Grigoriadis [6]. RMFGEN produces networks which consist of \( b \) frames. Each frame is a grid graph with side length \( a \). The frames are connected by \( a^2 \) forward and \( a^2 \) backward edges with random capacity. An \( a, b \)-RMFGEN network has \( a^2 b \) vertices and \( 6a^2 b - 4ab - 2a^2 \) edges. We generated 10 different networks with 45 up to 972 vertices.

All experiments were run on the HP 715/33 of the Abt. Math. Optimization at TU Braunschweig. Both implementations (plain and improved) were coded in ANSI-C. We used the preflow-push maxflow algorithm introduced by Goldberg and Tarjan [5] in an implementation of Derigs and Meier [2]. Here is a table with the computational results. \( \frac{u}{v} \) stands for \( u \) sec. for reduction and \( v \) sec. for splitting in the plain implementation and \( x, y \) sec. for the improved version.

<table>
<thead>
<tr>
<th>( k ) ((a, b))</th>
<th>(3,5)</th>
<th>(4,6)</th>
<th>(4,7)</th>
<th>(5,8)</th>
<th>(6,8)</th>
<th>(7,9)</th>
<th>(8,10)</th>
<th>(8,11)</th>
<th>(9,11)</th>
<th>(9,12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>0</td>
<td>65</td>
<td>1</td>
<td>104</td>
<td>1</td>
<td>704</td>
<td>4</td>
<td>2029</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>0</td>
<td>64</td>
<td>1</td>
<td>102</td>
<td>1</td>
<td>702</td>
<td>7</td>
<td>2238</td>
<td>17</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>0</td>
<td>39</td>
<td>1</td>
<td>95</td>
<td>1</td>
<td>664</td>
<td>7</td>
<td>2536</td>
<td>17</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0</td>
<td>57</td>
<td>1</td>
<td>92</td>
<td>1</td>
<td>622</td>
<td>7</td>
<td>2667</td>
<td>17</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>0</td>
<td>52</td>
<td>0</td>
<td>86</td>
<td>0</td>
<td>586</td>
<td>6</td>
<td>2015</td>
<td>16</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>0</td>
<td>45</td>
<td>0</td>
<td>70</td>
<td>0</td>
<td>588</td>
<td>2</td>
<td>1720</td>
<td>5</td>
</tr>
<tr>
<td>1000</td>
<td>3</td>
<td>0</td>
<td>39</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>459</td>
<td>2</td>
<td>1563</td>
<td>5</td>
</tr>
</tbody>
</table>

* \( \lambda_c > k \), \( \dag \): total running time > 5h.

It is clear from the table that the theoretical improvements, described in section 3 lead to improvements in running time.

## 5 Conclusions

In [4], more general problem definitions are given, e.g. networks with underlying undirected graphs, with pre-described capacities at some vertices, or minimum cost augmentations for which costs arise from vertex costs. In this paper we have restricted attention to the simple directed case, but we remark that our algorithmic improvements of section 3 also work for the more complex cases. The software and the technical report [1] can be obtained from the Opt-Net server elib.zib-berlin.de.

### References


