Improving time bounds on maximum generalised flow computations by contracting the network

Tomasz Radzik*

Department of Computer Science
King’s College London
London WC2R 2LS, United Kingdom
radzik@ics.kcl.ac.uk

Abstract. We consider the maximum generalised network flow problem and a supply-scaling algorithmic framework for this problem. We present three network-modification operations, which may significantly decrease the size of the network when the remaining node supplies become small. We use these operations in Goldfarb, Jin and Orlin’s supply-scaling algorithm and prove a $O(m^2n \log B)$ bound on the running time of the resulting algorithm. The previous best time bounds on computing maximum generalised flows were the $O(m^{1.5}n^2 \log B)$ bound of Kapoor and Vaidya’s algorithm based on the interior-point method, and the $O(m^2 \log B)$ bound of Goldfarb, Jin and Orlin’s algorithm.

1 Introduction

In a generalised flow network, each arc $e$ has a gain factor $\gamma(e)$ associated with it, and if $x$ units of flow enter $e$, then $\gamma(e)x$ units arrive at the other end. Each node has specified supply of one common commodity. The objective of the maximum generalised flow problem is to design flow which carries these node supplies through the network to one distinguished node, the sink. The designed flow must maximise the amount of commodity arriving at the sink and cannot violate the capacities of arcs. This problem models some optimisation problems arising in manufacturing, transportation and financial analysis [1–3].

The maximum generalised flow problem is a special case of linear programming, so it can be solved by any general-purpose linear programming method. The best asymptotic time bound on computing maximum generalised flows using this approach is the $O(m^{1.5}n^2 \log B)$ bound of Kapoor and Vaidya’s algorithm [4, 5] based on Karmarkar’s interior-point method. Here $n$ is the number of nodes, $m$ is the number of arcs, and $B$ is the largest integer in the representations of the capacities and gain factors of arcs and the supplies at nodes, assuming that these numbers are given as ratios of two integers.

The other line of research in designing generalised-flow algorithms follows the combinatorial approach to network flow problems originated by Ford and Fulkerson [6]. A combinatorial algorithm for the maximum generalised flow problem

* This work was supported by the UK EPSRC grant GR/L81468.
exploits the combinatorial structures of the underlying network and of the flows in this network, and often uses as subroutines combinatorial algorithms for simpler network problems, such as the shortest paths problem and the maximum (non-generalised) flow problem. The first polynomial-time bound on computing maximum generalised flows by combinatorial algorithms was shown by Goldberg, Plotkin, and Tardos [7], and the best, prior to our paper, bound of this type is $O(m^3 \log B)$, due to Goldfarb, Jin and Orlin [8].

Kapoor and Vaidya’s algorithm and Goldfarb, Jin and Orlin’s algorithm give the previous best asymptotic time bounds on computing maximum generalised flows. The main conclusion of our paper is a combinatorial algorithm which computes maximum generalised flows in $O(m^2 n \log B)$ time. This bound improves the previous bounds if $m = O(n^{1+\epsilon})$ and $m = O(n^{2-\epsilon})$, for any constant $\epsilon > 0$.

Goldfarb, Jin and Orlin [8] two algorithms which have the following supply-scaling structure. The computation consists of scaling phases. During the current phase, the remaining node supplies are sent in chunks of $\Delta$ units towards the sink along the highest gain paths. The scaling parameter $\Delta$ decreases at least by half at the end of each phase. Each phase has $O(m)$ iterations and each iteration is dominated by one single-source shortest-path computation. We present in this paper three network-modification operations, which are intended to decrease the size of the network during the computation of a supply-scaling algorithm. If the network does become smaller, then the subsequent phases may run faster. The first operation is a standard operation of contracting two nodes, if there is enough arc capacity between them (in both direction) to accommodate all remaining node supplies. The other two operations are an operation of by-passing (and removing) some nodes and an operation of shortcutting some paths. We believe that in practice these operations may significantly decrease the size of the network and speed-up the computation, but in this paper we focus on the question if they can lead to improved asymptotic time bounds.

The computation of Goldfarb, Jin and Orlin’s algorithms [8] terminates when the remaining node supplies total to less than $B^{-m}$, and an optimal flow is obtained by simple post-processing. The node supplies drop below $B^{-m}$ in $O(m \log B)$ phases, so the total running time is $O(m^3 \log B)$. We show that the size of the network must decrease when the remaining node supplies become $B^{-O(m)}$. More precisely, but abstracting from technical details, we show that if the node supplies are $B^{-O(m)}$, then our network-modification operations reduce the number of arcs to $O(m/k^2)$. This decreases the bound on the total running time of all $O(m \log B)$ phases by factor $m/n$ to our new bound of $O(m^2 \log B)$.

An essential tool in our analysis is a simple fact that the value of an expression composed of additions and multiplications of $k$ fractional numbers with denominators bounded by $B$ is a fractional number with the denominator bounded by $B^k$. The absolute value of such a number is either 0 or greater than $B^{-k}$. We use this fact in the following way. If the remaining supply at some node is still positive but less than $B^{-k}$, then $\Omega(k)$ arcs must “contribute” to the value of this.

1 Notation $O(\cdot)$ hides a factor polylogarithmic in $n$. 
remaining supply. We show a relation between the number of these contributing arcs and the decrement of the size of the network.

2 Definitions

A generalised flow network \( G = (V, E, t, \gamma, u, \delta) \) consists of: a set of nodes \( V \); a set of (directed) arcs \( E \); a sink node \( t \in V \); a gain function \( \gamma : E \rightarrow (0, \infty) \) \((\gamma(e) \) is the gain factor of arc \( e \)); a capacity function \( u : E \rightarrow [0, \infty] \); and a node supply function \( \delta : V \setminus \{t\} \rightarrow [0, \infty] \) \((\delta(v) \) is the initial supply at node \( v \)).

There may be multiple arcs in \( G \), and we will normally denote an arc from a node \( v \) to a node \( w \) by \( e_{v,w} \). We assume, without loss of generality, that all arcs are matched into pairs of reverse arcs. If an arc \( e' \) goes from a node \( v \) to a node \( w \) in \( V \), then its reverse arc \( e'' \) goes from \( w \) to \( v \) and \( \gamma(e'') = 1/\gamma(e') \).

Let \( E^- \) and \( E^+ \) denote the sets of arcs outgoing from \( v \) and incoming to \( v \), respectively. If there is an arc from a node \( v \) to a node \( w \), then we call the pair \( \{v, w\} \) an edge. We denote the set of all edges in \( G \) by \( E \). The gain factors are sometimes called loss/gain factors. For a path or a cycle \( P \) in \( G \), the gain factor of \( P \) is equal to \( \gamma(P) = \Pi \gamma(e) : e \in P \). A cycle \( P \) of positive-capacity arcs is called a flow-generating cycle or a flow-absorbing cycle or a 1-gain cycle, if \( \gamma(P) \) is greater, less, or equal to 1, respectively. If network \( G \) does not contain a flow generating cycle, then we call it a non-gain network.

Let \( n \) and \( m \) denote the number of nodes and the number of arcs in network \( G \). The arc gain factors and capacities and the node supplies are given as ratios of integers. We denote the set of these fractional input numbers by \( D_{\text{frac}} \), and the largest integer among the enumerators and denominators in \( D_{\text{frac}} \) by \( B \). To simplify asymptotic time bounds, we assume that \( B \geq n \). We also assume in this extended abstract that, despite allowing multiple arcs, \( m \leq n^2 \). See [9] for the technical details covering the case when \( m > n^2 \).

A (generalised) flow \( f : E \rightarrow (-\infty, +\infty) \) satisfies the following conditions.

1. Skew symmetry: for each pair of reverse arcs \( e' \) and \( e'' \), \( f(e'') = -\gamma(e')f(e') \).
2. Capacity constraint: for each arc \( e \in E \), \( f(e) \leq u(e) \).
3. Flow conservation: for each node \( v \in V \setminus \{t\} \), \( \sum_{e \in E^-} f(e) \leq \delta(v) \).

If \( f(e_{v,w}) \) units of flow enter arc \( e_{v,w} \) at node \( v \), then \( \gamma(e_{v,w})f(e_{v,w}) \) units arrive at node \( w \). The actual flow is defined by those flow values \( f(e) \) which are positive, while the negative flow values on the reverse arcs (Condition 1) are only for notational convenience. The sum in Condition 3 is the net-flow outgoing from node \( v \). The value of a flow \( f \) is the net-flow into the sink \( t \). The maximum generalised network flow problem is to compute a flow in a given network \( G \) with the maximum possible value. Such a flow is a maximum flow or an optimal flow.

For a flow \( f \), the residual capacity of an arc \( e \) is \( u_f(e) = u(e) - f(e) \), and the residual supply at a node \( v \) is \( \delta_f(v) = \delta(v) - \sum_{e \in E^-} f(e) \). The residual network of \( G \) with respect to a flow \( f \) is the network \( G_f = (V, E, t, \gamma, u_f, \delta_f) \). An optimal flow \( f_{\text{opt}} \) in the residual network \( G_f \) gives an optimal flow \( f_{\text{opt}} = f + f_{\text{opt}} \) in \( G \).
A labeling in \( G \) is a function \( \mu : V \rightarrow (0, \infty) \), \( \mu(t) = 1 \). The re-labeled network \( G_\mu = (V, E, t, \gamma_\mu, u_\mu, \delta_\mu) \) is network \( G \) “normalised” with labeling \( \mu \):

\[
\delta_\mu(v) = \delta(v)\mu(v), \quad u_\mu(e_{v,w}) = u(e_{v,w})\mu(v), \quad \gamma_\mu(e_{v,w}) = \gamma(e_{v,w})\mu(w)/\mu(v).
\]

If \( f \) is a flow in \( G \), then the same flow expressed in the re-labeled network \( G_\mu \) is denoted by \( f_\mu \), and \( f_\mu(e_{v,w}) = f(e_{v,w})\mu(v) \), for each \( e_{v,w} \in E \). We assume that for any flow \( f \) in \( G \), there are paths of positive residual-capacity arcs to the sink \( t \) from all other nodes. (One can modify network \( G \) into an equivalent network \( G' \) satisfying this condition, and \( n' = O(n), m' = O(m+n), B' = B \).) If a residual network \( G_f \) is a non-gain network, then the canonical labeling \( \mu \) of \( G_f \) is defined by the maximum gains of paths to the sink. That is, \( \mu(v) \) is equal to the maximum \( \gamma(P) \) over all positive residual-capacity paths \( P \) from \( v \) to \( t \). For this labeling \( \mu, \gamma_\mu(e) \leq 1 \), for each positive residual-capacity arc \( e \in E \), and we call network \( G_{f,\mu} \) a canonical residual network. If the gain of each positive residual-capacity arc is at most 1 (a common invariant in generalised flow algorithms), then the canonical labeling of \( G_f \) can be computed in \( O(m) \) time using Dijkstra’s shortest-paths algorithm. If \( f \) and \( \mu \) are a flow and a labeling in network \( G \), then the total re-labeled residual supply is defined as \( \Delta_{f,\mu} = \sum_{v \in V \setminus \{t\}} \delta_{f,\mu}(v) \).

If the capacities of a pair of reverse arcs \( e_{v,w} \) and \( e_{w,v} \) are both positive, then we call such arcs active arcs and the edge \( \{v, w\} \) an active edge. We call \( G \) a basic network, if it does not contain a cycle of active edges and does not contain a path of active edges between two nodes with positive supplies or between a node with positive supply and the sink \( t \). We call a flow \( f \) basic flow, if the residual network \( G_f \) is basic. We call a flow \( f \) a maximal flow, if the residual network \( G_f \) is a non-gain network. For an arbitrary generalised flow network \( G \), one can compute in \( O(mn^2 \log B) \) time a maximal flow \( f' \) [7]. Having a maximal flow \( f' \) and the canonical labeling \( \mu \) of \( G_f' \), one can compute in \( O(m) \) time a basic maximal flow \( f'' \) such that: the value of flow \( f'' \) is not less than the value of flow \( f' \); and the new total re-labeled residual supply \( \Delta_{f''} \) is not greater than the previous total re-labeled residual supply \( \Delta_{f'} \). Therefore we can assume that the computation always begins with a basic non-gain residual network.

The following theorem is proven in [7] (see also [8]).

**Theorem 1.** Let \( f \) be a maximal flow in network \( G \), and let \( \mu \) be the canonical labeling of \( G_f \). If \( \Delta_{f,\mu} < B^{-m} \), then a maximum generalised flow in network \( G \) can be obtained by one maximum non-generalised flow computation in the network \( G_{f,\mu} \) restricted to the arcs with re-label gain \( \gamma_{\mu} \) equal to 1.

If \( f \) is a maximal flow in \( G \) and \( \Delta_{f,\mu} < B^{-m} \), where \( \mu \) is the canonical labeling of \( G_f \), then we call \( f \) a near-optimal flow. Theorem 1 implies that having a near-optimal flow, we can compute an optimal one in \( O(mn) \) additional time, so from now on we assume that the objective is to compute a near-optimal flow.

### 3 An underlying approximation algorithm

Let \( \mathcal{A} \) be an iterative approximation algorithm for the maximum generalised flow problem which has the following properties. The input is a canonical non-gain
residual network $G_{f_r^{(k)}}$ and the computation consists of a sequence of phases. The input to a phase is a canonical non-gain residual network $G_{f^{(k)}}$ and the output (the input for the next phase) is another canonical non-gain residual network $G_{f_r^{(k+1)}}$. Each phase reduces the total residual supply at least by half: $\Delta f^{(k)} \leq \Delta f_r^{(k+1)}/2$. The computation ends when the total residual supply falls below the desired approximation threshold. The running time of one phase is at most $T(n, m)$, where the bounding function is such that $T(n, cm) \leq cT(n_2, m)$, if $n_1 \leq n_2$ and $c \leq 1$. The total supply $\Delta f_{f_r^{(k)}}$ in $G_{f_r^{(k)}}$ is at most $nB^n$, so algorithm $A$ computes a near-optimal flow in $O(m\log B)$ phases, or in $O(T(n, m)n \log B)$ total time. We prove in this paper the following main theorem.

**Theorem 2.** A maximum generalised flow algorithm $A$ which has the properties described above can be modified to an algorithm $A'$ which computes a near-optimal flow in $O(n^2 \log B) + O(n^2)$ time.

Goldfarb, Jin and Orlin [8] algorithms have the properties of a maximum generalised flow algorithm $A$ stated above, with $T(n, m) = O(m^2)$, so they compute near-optimal flows in $O(m^2 \log B)$ time. Theorem 2 implies that these algorithms can be modified to compute near-optimal flows in $O(m^2 n \log B)$ time, so we have the following main conclusion of our paper.

**Theorem 3.** A maximum generalised flow can be computed in $O(m^2 n \log B)$ time.

4 Reducing the size of network

Let $G$ be a basic non-gain network and let $\mu$ be the canonical labeling of $G$. We say that an arc $e \in E$ has large capacity, if $u_\mu(e) > \Delta_\mu$; otherwise the arc has small capacity. If an arc $e \in E$ has large capacity, then for any basic maximal flow $f$ in $G$ which does not send flow along 1-gain cycles, $f(e) < u(e)$ (since such a flow can be decomposed into flows along simple paths). Thus if we change the capacity of arc $e$ to infinity and compute a basic maximal flow $f'$ in the modified network $G'$, then this flow, after removal of flow from 1-gain cycles, is a basic maximal flow in network $G$. We assume that whenever large-capacity multiple arcs from a node $v$ to a node $w$ appear during the computation, we remove all of them except the one with the largest gain factor (and we remove all arcs reverse to the removed large-capacity arcs; their capacities must be zero). If the capacities of a pair of reverse arcs $e_{v,w}$ and $e_{w,v}$ are both large, then we call edge $\{v, w\}$ a large-capacity edge or a contractable edge. Observe that in such a case $\gamma_\mu(e_{v,w}) = \gamma_\mu(e_{w,v}) = 1$.

**Contracting large-capacity edges.** To contract a large-capacity edge $\{v, w\}$, where $w \neq t$ and $\delta(w) = 0$, means to create a new network $H_{\mu}$ by modifying network $G_{t}$ in the following way. For each arc adjacent to node $w$, replace the end node $w$ with node $v$ (the re-labeled gain $\gamma_\mu$ and capacity $u_\mu$ of this arc remain unchanged). Then after this changes remove node $w$. 
For a basic non-gain network $G$, let $\overrightarrow{F} \subseteq \overrightarrow{E}$ and $\overrightarrow{F} \subseteq \overrightarrow{E}$ be the forest of active edges and the forest of large-capacity edges, respectively. A component $C \subseteq V$ of network $G$ is a connected component of $(V, \overrightarrow{F})$, and a strong component $C \subseteq V$ is a connected component of $(V, \overrightarrow{F})$. In each tree of the forest $(V, \overrightarrow{F})$, there is at most one node which has positive supply or is the sink $t$ (see the definition of a basic network). Let $H$ be the basic non-gain network obtained from network $G$ by contracting all contractable edges. Network $H$ can be computed in $O(m)$ time. The strong components of network $G$ correspond to the nodes of network $H$. Every flow in network $G_\mu$ has the natural corresponding flow in network $H_\mu$. Conversely, a basic maximal flow in network $H_\mu$ can be expanded in $O(m)$ time into a basic maximal flow in network $G_\mu$ which has the same value and the same residual node supplies.

**Shortcutting small reverse-flow paths.** Let $G$ be a non-gain network and let $\mu$ be the canonical labeling of $G$. Let $P$ be a (directed) path from a node $v$ to a node $w$ such that the sink $t$ is not an intermediate node, the (re-labeled) capacities $u_\mu$ of all arcs on the path are the same, the capacities of all arcs reverse to the arcs on $P$ are large, and there are no positive supplies at the intermediate nodes (see Figure 1). We call such a path $P$ a small reverse-flow path because it would occur as the reverse residual path of a small flow from $w$ to $v$. Modify network $G$ by adding a new arc $e_{w,v}$ with capacity $u_\mu$ equal to the capacity $u_\mu$ of $P$ and with gain $\gamma_\mu$ equal to 1 (the same as the gain factors $\gamma_\mu$ of the arcs on $P$). Add also a zero capacity reverse arc $e_{w,v}$. Finally, set the capacities of the arcs on $P$ to zero. Observe, that if all positive-capacity arcs adjacent to the intermediate nodes on $P$ other than the arcs of $P$ have large capacities, then this shortcutting procedure makes all intermediate nodes on $P$ free.

A flow in the modified non-gain network $H$ can be easily converted into a flow of the same value and with the same residual node supplies in network $G$. Conversely, if we have a flow $f$ in $G$, then we can obtain a flow $h$ of the same value and with the same residual node supplies in network $H$ in the following way. Let $M = \max \{ 0, \max \{ f_\mu(e) : e \in P \} \}$, decrease the flow along $P$ by $M$ and send these $M$ units of flow along the added arc $e_{w,v}$. That is, set $h_\mu(e_{w,v}) = M$, $h_\mu(e) = f_\mu(e) - M$ and $h_\mu(e') = f_\mu(e') + M$, for every arc $e \in P$ and arc $e'$ reverse to arc $e$.

**By-passing and removing free nodes.** Let $G$ be a basic non-gain network. We say that a node $v \in V \setminus \{ t \}$ is free, if it does not have positive supply and for each pair of reverse arcs adjacent to $v$, the capacity of one of them is large and the capacity of the other is zero. Let $V_{\text{free}} \cup V_{\text{non-free}} = V$ be the partitioning of the set of nodes into the free nodes and non-free nodes. We can remove a free node $v \in V_{\text{free}}$ from the network in the following way. For each pair of positive (hence large) capacity arcs $e_{x,y}$ and $e_{y,z}$, add a new arc $e_{x,z}$ with infinite capacity and gain equal to $\gamma_\mu(e_{x,y}) \gamma_\mu(e_{y,z})$ (and the zero-capacity arc reverse to $e_{x,z}$). Then remove node $v$ and all arcs adjacent to it. In this way we obtain a basic non-gain network $H$ which, for our purpose, is equivalent to the original network $G$.

If we remove one or more free nodes, then up to 4 arcs may be added between any pair of the remaining nodes $x$ and $z$: one infinite-capacity arc from $x$ to $z$. 

one infinite-capacity arc from \( z \) to \( x \), and the arcs reverse to these two arcs. (Remember that we always remove all multiple large-capacity arcs but one.) Let \( \bar{n} = |V_{\text{non-free}}| \) and let \( m' \) denote the number of arcs which originate and end in non-free nodes. If we remove all free nodes from network \( G \), then we get a network with \( \bar{n} \) nodes and \( \bar{n} \leq m' + 2\bar{n}(\bar{n} - 1) \) arcs. For example, if the input network does not have multiple arcs, then \( m' \leq \bar{n}(\bar{n} - 1) \), so \( \bar{n} \leq 3\bar{n}(\bar{n} - 1) \). In such a case, if \( \bar{n} = o(\sqrt{m}) \), then \( \bar{n} = o(m) \).

Simultaneous removal of all free nodes amounts to adding a new infinite-capacity arc \( e_{x,y} \) for each pair of non-free nodes \( x \) and \( y \). The gain factor of this arc is equal to the largest gain of a path from \( x \) to \( y \) which passes only through free nodes. We can compute the gain factors of all these new arcs in \( O(\bar{n}m) \) time by applying Dijkstra’s single-source shortest-paths algorithm to each non-free node as the source. Let \( H \) denote the basic non-gain network obtained from network \( G \) by removal of all free nodes. If we have a basic maximal flow in network \( H \), and have recorded the \( \bar{n} \) shortest-path trees computed to set the new edges in \( H \) as mentioned above, then we can obtain a maximal flow of the same value and with the same residual node supplies in network \( G \) in \( O(\bar{n}m) \) time. (For each node \( x \) in \( H \), shift the flow from the added arc \( e_{x,y} \) onto the arcs of the recorded tree; update the flow on the arcs of the tree starting from the leaves for the running time of \( O(n) \) per one tree.)

**Procedure** *Shrink*\((G)\) modifies a basic non-gain network in the following way.

1. Contract all contractable edges in \( G \). Let \( G^{(1)} \) denote the obtained network.
2. Shortest every small reverse-flow path in \( G^{(1)} \) to obtain network \( G^{(2)} \).
3. Remove all free nodes from \( G^{(2)} \) to obtain network \( G^{(3)} \).
4. Return network \( G^{(3)} \) or the initial network \( G \), whichever has fewer arcs.

The running time of procedure *Shrink*\((G)\) is dominated by the \( \tilde{O}(nm) \) time required by step 3. A basic maximal flow in the shrunk network \( G^{(3)} \) can be expanded in \( O(nm) \) time to a maximal flow of the same value and with the same residual node supplies in network \( G \).

Let \( A \) be a maximum generalised flow algorithm which has the properties as described in Section 3. We modify such an algorithm into an algorithm \( A' \) by
periodically applying procedure \textsc{Shrink} to the current non-gain residual network. Before each application of procedure \textsc{Shrink}, we first modify the current residual network into a basic non-gain residual network. At the end of the computation of algorithm $A'$, the final flow in the final shrunk network is expanded to a flow of the same value in the initial network $G$. We prove now our main Theorem 2, using the following theorem (proven in Section 5).

**Theorem 4.** Let $G_f$ be a basic non-gain residual network, let $\mu$ be a canonical labeling of $G_f$, and let $1 \leq k \leq m/n$. There exists a constant integer $\alpha > 0$ such that if $\Delta_f,\mu \leq B^{3kn}$, then procedure $\textsc{Shrink}(G_f)$ returns a network with the number of nodes $n' \leq n$ and the number of arcs $m' \leq \min\{m, \lfloor (\alpha m)/(kn)^2 \rfloor \}$.

**Proof of Theorem 2.** We execute procedure \textsc{Shrink} every $cn \log B$ phases of algorithm $A$, for a suitably large constant $c$. We view the computation of this modified algorithm $A'$ as a sequence of stages. Stage 0 consists of the initial $cn \log B$ phases, and each subsequent stage consists of the execution of procedure \textsc{Shrink} followed by further $cn \log B$ phases of algorithm $A$. Since there are $O(m \log B)$ phases in algorithm $A$, there are $K = O(m/n)$ stages in algorithm $A'$.

To simplify the argument, we assume that procedure \textsc{Shrink} is not applied to the current, possibly already shrunk network to shrink it even further, but to a residual network of the initial input network $G$.

Let $f_k$ be the basic maximal flow in $G$ before the $k$-th application of \textsc{Shrink}, and let $\mu_k$ be the canonical labeling of $G_{f_k}$. The geometric decrease of the residual supply during the computation of $A$ implies that the total residual supply in network $G_{f_k,\mu_k}$ is $\Delta_{f_k,\mu_k} \leq nB^{\alpha \log B}/(kn \log B) \leq B^{-3kn}$. Let $n_k$ and $m_k$ be the numbers of nodes and arcs in the network computed by \textsc{Shrink}(G_{f_k}), that is, the size of the network used during stage $k$. Theorem 4 implies that $n_k \leq n$ and $m_k \leq \lfloor (\alpha m)/(kn)^2 \rfloor$. Hence $m_k \leq (\alpha/k^2)m$ for $k \geq \alpha$, assuming that $m \leq n^2$. The running time of algorithm $A'$ is $T(n, m)O(n \log B) + O(Knm)$ for the first $\alpha$ stages and all computations of procedure \textsc{Shrink}, plus

$$\sum_{k=\alpha}^{K} O(T(n_k, m_k) n \log B) \leq \sum_{k=\alpha}^{K} \frac{\alpha^2}{k^2} O(T(n, m) n \log B) \leq O(T(n, m) n \log B).$$

The first inequality follows from the property of the bounding function $T(n, m)$ stated in Section 3. The second inequality holds because $\sum_{i=1}^{\infty} 1/i^2 = \Theta(1)$.

**Remark 1.** The above proof uses the assumption that $m \leq n^2$, but can be extended to the case when $m > n^2$ by bounding separately the running time of the first $\Theta(m/n^2)$ stages and the running time of the remaining stages (see [9]).

5 Size of a strong component when supply is very small

In this section we prove Theorem 4. Define the degree $\deg(C)$ of a set of nodes $C \subseteq V$ as the number of arcs with at least one end in $C$. The proof is based on Lemmas 2, 3 and 4, which say that when the total residual supply is exponentially
small, then a strong component must have large degree. These three lemmas considers different types of strong components. Their proofs are similar, so we include only the proof of Lemma 2. The proofs are based on Lemma 1, which gives an expression on the balance of flow at a subset of nodes, and on the following observation. To obtain an exponentially small value from the fractional input numbers $D_{trac}$ using additions and multiplications, we have to use quite a few of them. We also need Lemma 5, which says that when the total residual supply is very small, at least one arc in each pair of reverse arcs must have large capacity.

A bi-directional tree $T \subseteq E$ spanning a set of nodes $C \subseteq V$, $C \neq \emptyset$, consists of $|C| - 1$ pairs of reverse arcs spanning $C$. The next lemma states the flow conservation property with respect to a tree.

**Lemma 1.** Let $T \subseteq E$ be a bi-directional tree spanning a non-empty set of nodes $C \subseteq V \setminus \{t\}$. Let $r$ be an arbitrary node in $C$, and for a node $v \in C$, let $P_v \subseteq T$ denote the tree path from node $v$ to the root $r$. If $f$ is a flow in network $G$, then the following flow-balance relation holds:

$$\sum_{v \in C} \delta_f(v) \gamma(P_v) - \sum_{v \in C} \delta(v) \gamma(P_v) + \sum_{v \in C} \sum_{e \in E_v \setminus T} f(e) \gamma(P_v) = 0. \quad (1)$$

**Proof (Idea).** From the definition of the residual supplies, we have

$$\delta_f(v) - \delta(v) + \sum_{e \in E_v} f(e) = 0, \text{ for each } v \in C. \quad (2)$$

Eliminate in system (2) terms $f(e)$, for all $e \in T$, to obtain Equation (1).

**Lemma 2.** Let $G_f$ be a basic non-gain residual network, let $\mu$ be the canonical labeling of $G_f$, and let $C$ be a strong component of $G_f$ with the following properties. There are no active non-contractable edges adjacent to $C$ and there is a positive residual supply at one node in $C$. Under these conditions, if a number $d$ is such that $d \geq n$ and $\Delta f_{\mu} \leq B^{-d}$, then the degree of set $C$ is at least $d$.

**Fig. 2.** A strong component of Lemma 2. Black nodes are in $C$, white nodes are adjacent to $C$. Only positive residual-capacity arcs are shown; the tree arcs in bold.
Proof. If \( r \) is the node in \( C \) with positive supply (a component of a basic non-gain network has at most one node with positive supply) and \( T \) is the bi-directional tree of active arcs spanning \( C \) (see Fig. 2), then Equation (1) becomes
\[
\delta_f(r) - \sum_{v \in C} \delta(v) \gamma(P_v) + \sum_{v \in C} \sum_{e \in E_v \setminus T} f(e) \gamma(P_v) = 0. \tag{3}
\]
Assume that the degree of \( C \) is less than \( d \). All arcs in the last sum in (3) are non-active because \( C \) is a component. Hence for each arc \( e \) in this sum, the flow \( f(e) \) on this arc is either at its upper, \( u(e) \), or at its lower, \( -u(e')\gamma(e') \), bound, where \( e' \) is the reverse arc to arc \( e \). Thus the left-hand side of (3) excluding \( \delta_f(r) \) involves only numbers from set \( D_{\text{trac}}: \delta(v), \) for each \( v \in C \) (\( |C| \) numbers); either \( u(e) \), or \( u(e') \) and \( \gamma(e') \), for each non-tree arc \( e \) with tail in \( C \), where \( e' \) is the reverse arc of \( e \) (at most \( 2(\deg(C) - 2(|C| - 1)) \) numbers); and \( \delta(v)\), for each arc \( e \in T \) which is in the direction towards the root node \( r \) (\( |C| - 1 \) numbers). Thus at most \( 2\deg(C) + 1 \leq 2d - 1 \) numbers from \( D_{\text{trac}} \) are involved in (3), so the absolute value of the left-hand side of (3) excluding \( \delta_f(r) \) is a fractional number with denominator less than \( B^{2d} \). We get contradiction since
\[
0 < \delta_f(r) = \delta_{f,\mu}(r)/\mu(r) \leq \Delta_{f,\mu}B^{n-1} < B^{-3d+1}n < B^{-2d}.
\]

Lemma 3. Let \( G_f \) be a basic non-gain residual network, let \( \mu \) be the canonical labeling of \( G_f \), and let \( C \in V \setminus \{t\} \) be a strong component of network \( G_f \) with the following properties. There is exactly one active but non-contractable edge adjacent to \( C \), and there is no residual supply in \( C \). Under these conditions, if \( d \geq n \) and \( \Delta_{f,\mu} \leq B^{-3d} \), then the degree of set \( C \) is at least \( d \).

Fig. 3. Strong components of Lemmas 3 and 4.

Lemma 4. Let \( G_f \) be a basic non-gain residual network, let \( \mu \) be the canonical labeling of \( G_f \), and let \( C \subseteq V \setminus \{t\} \) be a strong component of \( G_f \) with the following properties. There is no residual supply in \( C \), and exactly two active but non-contractable edges \( \{x, p\} \) and \( \{y, q\} \) are adjacent to \( C \), where \( p, q \in C \). Let \( e_{x,p} \) and \( e_{p,x} \), and \( e_{y,q} \) and \( e_{q,y} \), be the two pairs of active reverse arcs. If \( d \geq n \) and \( \Delta_{f,\mu} \leq B^{-3d} \), then at least one of the following two conditions must hold:
(a) the degree of set $C$ is at least $d$, \\
(b) $u_{f,\mu}(e_{p,q}) = u_{f,\mu}(e_{y,q}) \leq \Delta_{f,\mu}$ or $u_{f,\mu}(e_{x,y}) = u_{f,\mu}(e_{x,y}) \leq \Delta_{f,\mu}$.

**Lemma 5.** Let $G_f$ be a basic non-gain residual network and let $\mu$ be the canonical labeling of $G_f$. If $\Delta_{f,\mu} \leq B^{-(n+1)}$, then at least one arc in every pair of reverse arcs in $G_{f,\mu}$ has large residual capacity.

**Proof.** Each arc with non-positive flow $f$ has large capacity.

**Proof of Theorem 4.** We apply procedure Shrink to the basic non-gain residual network $G_f$ given in the statement of the theorem. Let $G_f^{(1)}$, $G_f^{(2)}$ and $G_f^{(3)}$ denote the networks at the end of steps 1, 2 and 3 of procedure Shrink, respectively. Call a strong component of network $G_f$ large, if its degree is at least $kn$, and small otherwise. The nodes of $G_f^{(3)}$ correspond to some strong components of $G_f$. Call a node of $G_f^{(3)}$ large or small, depending whether it corresponds to a large or small strong component of $G_f$. The core of the proof is to show that $G_f^{(3)}$ has $O(m/(kn))$ nodes. We show this by showing that $G_f^{(3)}$ cannot have more small nodes than the large ones. Let $C$ be a component of $G_f$ and estimate how many small and large nodes it contributes to the final network $G_f^{(3)}$.

If $C$ consists of only one strong component and contains the sink $t$, then it becomes the sink node in $G_f^{(3)}$, and it may be either large or small. If $C$ consists of only one strong component, does not contain the sink $t$, and does not have any residual supply, then it is contracted into a single free node in $G_f^{(1)}$ and then removed. If $C$ consists of only one strong component but does have residual supply, then Lemma 2 implies that $C$ is a large component, so it contributes to $G_f^{(3)}$ only one node and this node is large. For the remaining case, when $C$ consists of at least two strong components, let $W$ be the set of nodes of $G_f^{(3)}$ contributed by $C$, and let $T$ be the tree in $G_f^{(3)}$ of the active edges spanning $W$. One node in $W$ may either be the sink or have positive residual supply; let $W^t \subseteq W$ denote the set of the other nodes in $W$. Lemma 3 implies that a node in $W^t$ which is a leaf of tree $T$ must be large. Lemma 4 implies that a node in $W^t$ of degree 2 in $T$ must be also large (otherwise it is shortcut and removed). Thus a small node in $W^t$ has at least degree 3 in $T$. There are at most $|W|/2 - 1$ nodes in $T$ of degree at least 3, so there are at most $|W|/2$ small nodes in $W$.

Let $\bar{n}$ denote the number of nodes in the final network $G_f^{(3)}$. We have shown that at least $\bar{n}/2 - 1$ nodes in $G_f^{(3)}$ are large, so at least that many strong components of $G_f$ are large. Thus there must be at least $kn(\bar{n}/2 - 1)/2$ arcs in network $G_f$ which are adjacent to strong components. Hence $m \geq kn(\bar{n}/2 - 1)/2$ and $\bar{n} \leq 5m/(kn)$ (assuming $\bar{n} \geq 10$). Lemma 5 implies that there are at most 4 arcs between any two nodes in network $G_f^{(3)}$: a small-capacity arc in one direction, a parallel large-capacity arc (with a smaller gain factor), and the two reverse arcs. Hence the number of arcs in network $G_f^{(3)}$ is at most $2(\bar{n})^2 \leq O(m^2/(kn)^2)$. 
6 Further questions

Tardos and Wayne [10] proposed a simple combinatorial algorithms for the maximum generalised flow problem, including a generalisation of Goldberg and Tarjan’s push-relabel algorithm for the min-cost flow problem [11]. Can our network modifications improve the time bounds of those algorithms by factor \( m/n \)?

Wayne [12] showed a polynomial-time combinatorial algorithm for the min-cost generalised flow problem. As in the maximum generalised flow case, the computation of this algorithm ends when the value of flow is within \( B^{-D(m)} \) from optimal. Can some network-modification operations work in this algorithm?

And finally, do the network-modification operations presented in this paper bring us any closer to settling the big open question of computing maximum generalised flows in strongly-polynomial time?

References