Experimantal evaluation of algorithmic solutions for generalised network flow models *

(Preliminary report)

Tomasz Radzik† and Shengxiang Yang†

King’s College London

July 2000

Abstract

The generalised network flow problem is to maximise the net flow into a specified sink node in a network with gain-loss factors associated with edges. In practice, computation of solutions for instances of this problem is almost always done using general-purpose linear programming codes, but this may change because a number of specialised combinatorial generalised-flow algorithms have been recently proposed. To complement the known theoretical analyses of these algorithms, we develop their implementations and investigate their practical performance. We include in our study different versions of Goldberg, Plotkin, and Tarjan’s Fast-Path algorithm and Wayne’s Push-Relabel algorithm. We compare the performance of our implementations of these algorithms with implementations of the straightforward highest-gain path-augmentation algorithms. We use various classes of networks, including a type of layered networks which may appear in the multiperiod portfolio revision problem.

Keywords: Network optimisation, Network flow algorithms, Generalised flow, Experimental evaluation.

1 Introduction

2 Definitions

The input

An instance, or an input, of the generalised flow problem is an antisymmetric generalised flow network \( \mathcal{G} = (V, \overline{E}, t, e, u, \gamma) \) where:

- \((V, \overline{E})\) is a directed graph with a set of nodes \( V \) and an antisymmetric set of edges \( \overline{E} \); that is, if \((v, w) \in \overline{E}\), then \((w, v) \notin \overline{E}\);
- \( t \in V \) is the sink, the destination of flow;
- \( e : V \to R_{\geq} \) is an (initial) excess (supply) function; assume that \( e(t) = 0 \);
- \( u : \overline{E} \to R_{\geq} \) is a capacity function;
- \( \gamma : \overline{E} \to R_{>0} \) is a gain (loss) function.

The assumption that the set of edges is antisymmetric is made without loss of generality. Symbols \( R_{\geq} \) and \( R_{>0} \) stand for the nonnegative and the positive real numbers, respectively. The meaning of the gain function \( \gamma \) is that if \( x \) units of flow enter an edge \((v, w)\) at node \( v \), then \( \gamma(v, w)x \) units arrive at node \( w \).

*This work is supported by the EPSRC grant GR/L81468.
†Authors’ address: Department of Computer Science, King’s College London, Strand, London WC2R 2LS, UK. E-mail: {radzik, yangs}@dcs.kcl.ac.uk
A flow and the optimisation objective

A flow in $\tilde{G}$ is a nonnegative function $f : \tilde{E} \rightarrow R_\geq$ which satisfies the flow conservation constraints (2): the net flow outgoing from each node $v$ cannot be greater than the initial excess at this node; and the capacity constraints (3): the amount of flow outgoing from a node $v$ along an edge $(v,w)$ cannot be greater than the capacity of this edge. The objective of the generalised flow problem is to find a flow which maximises the net flow incoming into the sink $t$. Thus the generalised flow problem is the following linear program with decision variables $\{f(v,w) : (v,w) \in \tilde{E}\}$.

\[(P) \quad \text{maximise:} \quad \sum_{(z,t) \in \tilde{E}} \gamma(z,t) f(z,t) - \sum_{(t,x) \in \tilde{E}} f(t,x); \quad (1)\]

subject to: \[\sum_{(v,z) \in \tilde{E}} f(v,z) - \sum_{(z,v) \in \tilde{E}} \gamma(z,v) f(z,v) \leq e(v), \quad \text{for each } v \in V - \{t\}; \quad (2)\]

\[f(v,w) \leq u(v,w), \quad \text{for each } (v,w) \in \tilde{E}; \quad (3)\]

\[f(v,w) \geq 0, \quad \text{for each } (v,w) \in \tilde{E}. \quad (3)\]

The first sum in (2) is the flow outgoing from node $v$ and the second one is the flow incoming into $v$.

The dual problem

Let $\mu(v)$, for $v \in V - \{t\}$, and $\lambda(v,w)$, for $(v,w) \in \tilde{E}$, be the dual variables associated with constraints (2) and (3). Introducing a constant $\mu(t) = 1$ (to simplify the formulation) we obtain the following dual problem.

\[(D) \quad \text{minimise:} \quad \sum_{v \in V} e(v) \mu(v) + \sum_{(v,w) \in \tilde{E}} u(v,w) \lambda(v,w); \quad (4)\]

subject to: \[\mu(v) - \gamma(v,w) \mu(w) + \lambda(v,w) \geq 0, \quad \text{for each } (v,w) \in \tilde{E}; \quad (5)\]

\[\mu(v) \geq 0, \lambda(v,w) \geq 0, \quad \text{for each } v \in V \text{ and each } (v,w) \in \tilde{E}; \quad (6)\]

\[\mu(t) = 1. \quad (7)\]

For a fixed feasible vector $\mu$, the objective function of (D) is minimised by setting

\[\lambda_\mu(v,w) = \max \{0, \gamma(v,w) \mu(w) - \mu(v)\}, \quad \text{for each } (v,w) \in \tilde{E}. \quad (4)\]

A technical issue

A symmetric generalised flow network $H = (V,F,t,e,u,\gamma)$ has a symmetric set of edges $F$ (that is, if $(v,w) \in F$, then also $(w,v) \in F$), and $\gamma(v,w) = 1/\gamma(w,v)$, for each $(v,w) \in F$. For such a symmetric network, the generalised flow problem is formulated in the following way.

\[(P_s) \quad \text{maximise:} \quad \sum_{(z,t) \in \tilde{F}} \gamma(z,t) f(z,t); \quad (8)\]

subject to: \[f(w,v) = -\gamma(v,w) f(v,w), \quad \text{for each } (w,v) \in F; \quad (9)\]

\[\sum_{(v,x) \in \tilde{F}} f(v,x) \leq e(v), \quad \text{for each } v \in V - \{t\}; \quad (10)\]

\[f(v,w) \leq u(v,w), \quad \text{for each } (v,w) \in F. \quad (11)\]

Only positive edge flows are “real” flows. Symmetric edges and condition (5) are used to simplify the notions of residual paths and residual networks. In the case of an antisymmetric network, we have to refer to “sending flow along an edge $(v,w)$,” as well as to “reversing some of the flow sent before along $(v,w)$.” In the case of a symmetric network, we can refer only to “sending flow along an edge,” either $(v,w)$ or its symmetric $(w,v)$.

2
For our antisymmetric input network $\bar{G} = (V, \bar{E}, t, e, u, \gamma)$, let $G$ denote the symmetric network $(V, E, t, e, u, \gamma)$, where $E$ is obtained from $\bar{E}$ by adding to $\bar{E}$ the symmetric edge $(w, v)$ for each edge $(v, w) \in \bar{E}$, and setting $u(w, v) = 0$ and $\gamma(w, v) = 1/\gamma(v, w)$. For a flow $f$ in $\bar{G}$, the corresponding flow in $G$ is obtained by extending $f$ according to (5). Problem (P) for network $\bar{G}$ is the same as problem (P)$_s$ for network $G$, and from now on, for convenience, we refer only to symmetric networks. (We began with antisymmetric networks to formulate the generalised flow problem in a more natural way and to have an immediate derivation of the dual problem.)

**Residual network**

Let $f$ be a flow in network $G$. The residual capacities of edges in $E$ and the residual excesses of nodes in $V$ are defined in the following way.

\[
    u_f(v, w) = u(v, w) - f(v, w), \quad \text{for each } (v, w) \in E;
\]

\[
    e_f(v) = e(v) - \sum_{(v, w) \in E} f(v, w), \quad \text{for each } v \in V.
\]

Note that the sum above is the net flow outgoing from node $v$, since it covers the outgoing edges as well as the incoming ones (through their symmetric edges). A residual edge is an edge in $E$ with positive residual capacity $y$ and a residual path is a path which consists only of residual edges. Let $V_f$ denote the set of nodes which can reach the sink $t$ along residual paths, and let $E_f = E \setminus V_f \times V_f$ be the set of the edges between the nodes in $V_f$. The residual network is the generalised flow network $G_f = (V_f, E_f, t, e_f, u_f, y_f)$, with functions $e_f, u_f$, and $y$ restricted to sets $V_f$ and $E_f$. If $e_f(v) > 0$ for a node $v \in V_f$, then we say that $v$ is a node with (residual) excess.

If $h$ is a flow in $G_f$, then $f + h$ is a flow in $G$ creating the excess $e_f(t) + e_h(t)$ at the sink. (Assume that $h(v, w) = 0$ for each $(v, w) \in E - E_f$.) If $h$ is a maximum flow in $G_f$, then $f + h$ is a maximum flow in $G$.

Let $e_{\text{opt}}(t)$ denote the maximum possible net flow into the sink over all flows in $G$. We say that a flow $f$ in $G$ is $\xi$-optimal, if $e_f(t)(1 + \xi) \geq e_{\text{opt}}(t)$.

**Flow generating cycles**

The gain $\gamma(P)$ of a path $P$ is the product of the gains of the edges on this path. Sending $x$ units of flow from a node $v$ along a residual path $P$ to the sink increases the residual excess at the sink by $\gamma(P)x$.

A flow generating cycle is a residual cycle with gain greater than 1. If $\Gamma$ is a flow generating cycle, then we can increase flows on the edges of $\Gamma$ to create (or increase) the excess at at least one node on this cycle. This additional created excess can be sent subsequently to the sink.

Let $f$ be a flow in $G$ and $h$ a flow in the residual network $G_f$ such that the residual network $G_{f+h}$ of the combined flow $f + h$ does not have flow generating cycles. Such a flow $h$ cancels (all) flow generating cycles in the residual network $G_f$, and computation of $h$ is called canceling (all) flow generating cycles.

**Relabeled network**

Let $\mu$ be positive node labels in $G_f$, the relabeled (residual) capacities, the relabeled gain factors, and the relabeled (residual) excesses are defined as

\[
    u_{f, \mu}(v, w) = u_f(v, w)\mu(v),
\]

\[
    \gamma_{\mu}(v, w) = \gamma(v, w)\mu(w)/\mu(v),
\]

\[
    e_{f, \mu}(v) = e_f(v)\mu(v).
\]

The relabeled network $G_{f, \mu} = (V_f, E_f, s, u_{f, \mu}, e_{f, \mu}, \gamma_{\mu})$ is equivalent to network $G_f$ (they are equivalent instances of the generalised flow problem). For a flow $h$ in $G_f$, define the flow $h_{\mu}$ in $G_{f, \mu}$:

\[
    h_{\mu}(v, w) = h(v, w)\mu(v), \quad \text{for each } (v, w) \in E_f.
\]
\{ Input: a generalised flow network $G$ \}
\begin{align*}
    f &\leftarrow \text{a flow which cancels all flow generating cycles in } G; \\
    \textbf{while} & \text{ there is a node } v \in V_f \text{ with positive residual excess } \textbf{do} \\
    & 1. \text{ compute a highest gain path } P \text{ from } v \text{ to } t \text{ in } G_f; \\
    & 2. \text{ update flow } f \text{ by sending flow from } v \text{ to } t \text{ along } P \\
    & \quad \text{(transferring as much of the excess from } v \text{ to } t \text{ as the edge capacities on } P \text{ allow);} \\
    \textbf{end while} \\
    \textbf{output:} \text{ optimal flow } f \text{ in } G \\
\end{align*}

Figure 1: Onaga’s algorithm (code HIGHEST_PATH).

Flows $h$ and $h_\mu$ are actually the same flow, but $h$ expresses that flow in terms of the network $G_f$ while $h_\mu$ expresses it in terms of the equivalent network $G_{f,\mu}$. For a path $P$ from $v$ to $w$, $\gamma_\mu(P) = \gamma(P)\mu(v)/\mu(w)$, and for a cycle $\Gamma$, $\gamma_\mu(\Gamma) = \gamma(\Gamma)$.

There are no flow generating cycles in $G_f$ if and only if there exist node labels in $G_f$, called \textit{proper node labels}, such that the relabeled gain of each residual edge in $G_f$ is at most 1. If $G_f$ does not have flow generating cycles, then the node labels $\mu$ in $G_f$ such that $\mu(v)$ is equal to the highest gain of a path from $v$ to $t$ in $G_f$ are well defined, are proper, and are called the \textit{canonical node labels}. The canonical node labels, and the highest-gain paths to the sink, can be computed in $O(mn)$ time using the Bellman-Ford-Moore shortest paths algorithm. If we have proper node labels, then this computation can be done in $O(m + n \log n)$ time using Dijkstra’s shortest paths algorithm with Fibonacci heaps [?]. (Adapt shortest-path algorithms by replacing summing the edge weights with multiplying the edge gains.)

\textbf{Optimality conditions}

\textbf{Lemma 1.} If $f$ is a flow in $G$, then the following four conditions are equivalent.

1. Flow $f$ is optimal.
2. (a) There is no path from a node with excess to the sink $t$ in the residual network $G_f$, and (b) there is no flow generating cycle in the residual network $G_f$.
3. (a) There is no path from a node with excess to the sink $t$ in the residual network $G_f$, and (b) there exist node labels $\mu$ in $G_f$ such that the relabeled gain of each residual edge is at most 1.
4. There exist node labels $\mu$ in $G$ such that the feasible primal vector $f$ and the feasible dual vector $(\mu, \lambda_\mu)$ for the primal and the dual problems $(P)$ and $(D)$, satisfy the complementary slackness property ($\lambda_\mu$ is defined by (4)).

\textbf{Proof.}

(1) $\Rightarrow$ (2). If not (2) or not (3), then we would be able to send more flow into the sink.

(2) $\Rightarrow$ (3). Discussed above.

(3) $\Rightarrow$ (4). Extend the canonical node labels $\mu$ in $G_f$ to $V$ by setting $\mu(v) = 0$ for each $v \in V - V_f$, and check that vectors $f$ and $(\mu, \lambda_\mu)$ satisfy the complementary slackness property for the primal-dual pair (P) and (D).

(4) $\Rightarrow$ (1). From the complementary slackness optimality conditions. ■
\[
\{ \text{INPUT: a generalised flow network } G \} \\
f \leftarrow \text{a flow which cancels all flow generating cycles in } G; \\
\textbf{while} \quad \text{there is a node } v \in V_f \text{ with positive residual excess} \quad \textbf{do} \\
\quad 1: \text{compute the canonical node labels } \mu \text{ in } G_f \\
\quad 2: \text{compute a maximum flow } h \text{ in } G_f \mu \text{ from the nodes with excesses to } t \\
\quad \quad \text{using only edges with (relabelled) gains equal to } 1; \\
\quad 3: \quad f \leftarrow f + h; \\
\quad \{ \text{invariant: there are no flow generating cycles in } G_f \} \\
\textbf{end while} \\
\{ \text{OUTPUT: optimal flow } f \text{ in } G \}
\]

Figure 2: Truemper’s algorithm (code MAXFlow).

\[
\{ \text{INPUT: a generalised flow network } G \text{ and a number } \xi > 0 \} \\
\Delta \leftarrow \text{initialise;} \quad \{ \Delta \leq e_{\text{opt}}(t) \leq m\Delta \} \\
f \leftarrow \text{zero flow in } G; \\
\textbf{while} \quad m\Delta > \xi \cdot e_f(t) \quad \textbf{do} \\
\quad \{ e_{\text{opt}}(t) - e_f(t) \leq m\Delta \} \\
\quad h \leftarrow \text{a flow which cancels all flow generating cycles in } G_f; \\
\quad f \leftarrow f + h; \\
\quad \textbf{while} \quad \text{there is a } \Delta\text{-fat path from an excess node } v \text{ to } t \text{ in } G_f \quad \textbf{do} \\
\quad \quad \text{update flow } f \text{ by sending flow from } v \text{ to } t \text{ along a highest gain } \Delta\text{-fat path;} \\
\quad \textbf{end while} \\
\quad \{ e_{\text{opt}}(t) - e_f(t) \leq m\Delta/2 \} \\
\Delta \leftarrow \Delta/2; \\
\textbf{end while} \\
\{ \text{OUTPUT: } \xi\text{-optimal flow } f \text{ in } G \}
\]

Figure 3: The Fat-Path algorithm.
{ INPUT: a generalised flow network G and a number $\xi > 0$ }  
\begin{align*}  
f &\leftarrow \text{a flow which cancels all flow generating cycles in } G;  
\mu &\leftarrow \text{the canonical labeling in } G_f;  
\end{align*}
\begin{align*}  
\text{let } \text{TotResEx}_{\nu} &\text{ stand for the current sum of the residual relabeled excesses in } V_f - \{t\};  
\text{TotResEx}_{\nu} &\text{ is an upper bound on } e_{\text{opt}}(t) - e_f(t) \}  
\end{align*}
\begin{align*}  
\text{while } \text{TotResEx}_{\nu} &> \xi \cdot e_f(t) \ do  
\Delta &\leftarrow \text{TotResEx}_{\nu}(2(m+n));  
\text{Phase}(\Delta);  
\{ \text{ no flow generating cycles in } G_f, \mu - \text{the canonical labeling of } G_f, \text{TotResEx}_{\nu} < (m+n)\Delta \}  
\end{align*}
\begin{align*}  
\text{end while}  
\end{align*}
\begin{align*}  
\{ \text{output: } \xi\text{-optimal flow } f \text{ in } G \}.  
\end{align*}

\textbf{Phase}(\Delta):
\begin{align*}  
\{ \text{no flow generating cycles in } G_f, \mu - \text{the canonical labeling of } G_f \}  
\end{align*}
\begin{align*}  
\text{for each } v \in V_f - \{t\} \text{ and } (v,w) \in E_f \ do \ \hat{e}_{f,\nu}(v) &\leftarrow e_{f,\nu}(v); \ \hat{e}_{f,\nu}(v,w) &\leftarrow 0; \ \text{end for}  
\end{align*}
\begin{align*}  
\text{while } \text{there exists } v \in V_f - \{t\} \text{ such that } \hat{e}_{f,\nu}(v) &\geq \Delta \ do  
\text{PushFlowInPath}(v);  
\mu &\leftarrow \text{the canonical labeling in } G_f;  
\end{align*}
\begin{align*}  
\text{end while.}  
\end{align*}

\textbf{PushFlowInPath}(v):
\begin{align*}  
\text{while } v \neq t \text{ and } \hat{e}_{f,\nu}(v) &\geq \Delta \ do  
w &\leftarrow \text{the next node from } v \text{ on the highest gain path to } t;  
\text{PushFlowInEdge}(v,w);  
v &\leftarrow w;  
\end{align*}
\begin{align*}  
\text{end while.}  
\end{align*}

\textbf{PushFlowInEdge}(v,w):
\begin{align*}  
\text{do the following updates in } G_{f,\nu} \text{ (note that } \gamma_{\nu}(v,w) = 1) :  
moves \Delta \text{ units (of flow) from } \hat{e}_{f,\nu}(v) \text{ to } \hat{e}_{f,\nu}(v,w);  
send } \alpha &\leftarrow \min\{ \hat{e}_{f,\nu}(v,w), u_{f,\nu}(v,u) \} \text{ units of flow along } (v,w), \text{ increasing } \hat{e}_{f,\nu}(w,v) \text{ by } \alpha;  
moves } \min\{ \hat{e}_{f,\nu}(w,v), \Delta \} \text{ units from } \hat{e}_{f,\nu}(w,v) \text{ to } \hat{e}_{f,\nu}(w).  
\end{align*}

Figure 4: Goldfarb, Jin and Orlin’s excess scaling algorithm.

3 Onaga’s and Truemper’s algorithms

4 Excess scaling algorithms

4.1 The Fat-Path algorithm

4.2 Goldfarb, Jin and Orlin’s excess scaling algorithms

\textbf{EXSO:} implementation of the algorithm described in Figure 4.

\textbf{EXSC1:} Each \text{PushFlowInPath}(v) \ computation sends $k\Delta$ units of flow from node $v$ to the sink $t$ along the highest gain residual path, for the largest possible integer $k$.

\textbf{EXSC2:} Each \text{PushFlowInEdge}(v,w) \ computation sends $k\Delta$ units of flow from node $v$ along edge $(v,w)$ to node $w$, for the largest possible integer $k$.

\textbf{EXSC3:} During the computation \textbf{Phase}(\Delta), we apply the \text{PushFlowInPath}(v) \ computation to each node $v$ with excess $\hat{e}_{f,\nu}(v) \geq \Delta$ before computing the canonical labeling, and the new highest-gain paths tree. See Figure 4. \ Note next($v$) is the next node on the path from $v$ to $t$ in the last computed highest-gain paths tree. Procedure \text{PushFlowInPath}(v) \ may now terminate not only when the sink has been reached or the current node has excess $\hat{e}_{f,\nu}(v)$ less than $\Delta$, but also when an edge $(x,\text{next}(x))$ which no
**Phase**($\Delta$):

\{ no flow generating cycles in $G_f$, $\mu$ - the canonical labeling of $G_f$ \}

for each $v \in V_f - \{t\}$ and $(v, w) \in E_f$ do $\tilde{e}_{f, \mu}(v) \leftarrow e_{f, \mu}(v)$; $\tilde{e}_{f, \mu}(v, w) \leftarrow 0$ end for

while there exists $v \in V_f - \{t\}$ such that $\tilde{e}_{f, \mu}(v) \geq \Delta$ do

$L \leftarrow \{v \in V_f - \{t\} : \tilde{e}_{f, \mu}(v) \geq \Delta\};$

for each $v \in L$ do PUSHFLOWINPATH($v$) end for

$\mu \leftarrow$ the canonical labeling in $G_f$

end while.

PUSHFLOWINPATH($v$):

while $v \neq t$, $\tilde{e}_{f, \mu}(v) \geq \Delta$, and $u_f(v, next(v)) > 0$ do

PUSHFLOWINEDGE($v$, next($v$));

$v \leftarrow$ next($v$);

end while.

Figure 5: Modification of Goldfarb, Jin and Orlin’s algorithm (code ExSc3).

longer is residual has been reached. Procedure PUSHFLOWINEDGE($v, w$) is as in ExSc2.

5 Push-relabel algorithms

5.1 Single-phase push-relabel algorithm

5.2 Multi-phase push-relabel algorithm

6 Generators of generalised flow networks

To create test inputs for our implementations, we use two input generators of generalised flow networks. They are loosely based on possible applications of generalised flow problem in financial analysis. To simplify our initial tests, the generated networks do not have flow generating cycles.

Layered networks: multiperiod network portfolio models

Generator LAYERS creates a network with the set of nodes $V = \{(b, t) : a = 1, 2, \ldots, K; \tau = 1, 2, \ldots, T\} \cup \{t\}$. A node $(b, t)$ represents asset $b$ at the time period $\tau$. An edge $e = ((b', \tau), (b'', \tau + 1))$ represents possibility of exchanging an amount $x$ of asset $b'$ available at the time period $\tau$ into the $\gamma(e) \cdot x$ amount of asset $b''$, which will be available at the next time period $\tau + 1$. For each node $(b, \tau)$, $\tau < T$, generator LAYERS randomly chooses $D$ such edges outgoing from this node to nodes at the next time period. Each node at the last time period is connected by an edge to the sink $t$.

Generator LAYERSX is similar to generator LAYERS, but the edges can go from one time period to any other time period. A “forward” edge, from a time period $\tau'$ to a time period $\tau'' > \tau'$ represents exchanging assets (with possible delayed availability of the bought assets). A “backward” edge, from a time period $\tau'$ to a time period $\tau'' < \tau'$ represents borrowing.

The gains and the capacities of the edges are randomly selected from intervals $[\gamma_{\text{min}}, \gamma_{\text{max}}]$ and $[u_{\text{min}}, u_{\text{max}}]$, respectively. It seems that in this model one should expect the edge gains to be close to 1 (assuming that the data comes in a “normalised” form), so our intention is to set the parameters $\gamma_{\text{min}}$ and $\gamma_{\text{max}}$ close to 1.

Grid of cliques: exchanging and transferring currencies

The generator GRIDOFCLAIRES creates a network with the set of nodes $V = \{(c, q) : c = 1, 2, \ldots, K; q = 1, 2, \ldots, Q\} \cup \{t\}$, where a node $(c, q)$ represents currency $c$ at market $q$. For each $q = 1, 2, \ldots, Q$, the nodes $\{(c, q) : c = 1, 2, \ldots, K\}$ form a (directed) clique; and for each $c = 1, 2, \ldots, K$, the nodes
\[
\{ \text{input: a generalised flow network } G \text{ and a number } \xi > 0 \} \\
f \leftarrow \text{zero flow in } G; \\
\text{loop} \\
\quad \text{cancel all flow generating cycles in } G_f \text{ and compute the canonical labeling } \mu; \\
\quad \text{if } \text{TotResEx} \geq \xi e_f(t) \text{ then terminate;} \\
\quad h \leftarrow \text{Phase}(G_f, \mu, 1/2); \\
\quad f \leftarrow f + h; \\
\text{end_loop} \\
\{ \text{output: } \xi \text{-optimal flow } f \}. \\
\text{Phase}(G, \xi): \\
\{ G \text{ is a generalised flow network with edge gains of residual edges at most 1 } \} \\
\quad b \leftarrow (1 + \xi)^{1/n}; \\
\quad \text{round edge gains in } G \text{ to powers of } b \text{ and let } H \text{ be the resulting network;} \\
\quad h(v, w) \leftarrow 0, \text{ for each } (v, w) \text{ in } H; \\
\quad \mu(v) \leftarrow 0, \text{ for each } v \text{ in } H; \\
\text{while } \exists v \in H_h \text{ with positive excess do} \\
\quad \text{if exists an admissible edge } (v, w) \{ \text{a residual edge in } H_h \text{ with } \gamma_\mu(v, w) > 1 \} \text{ then} \\
\quad \quad \text{push: } \text{update } f \text{ by sending } \min \{e_h(v), u_h(v, w)\} \text{ units of flow along } (v, w); \\
\quad \quad \text{else} \\
\quad \quad \quad \text{relabel: } \mu(v) \leftarrow \mu(v)/b^{1/n}; \\
\text{end_while} \\
\{ h \text{ is an optimal flow in } H \} \\
\quad g \leftarrow \text{interpretation of flow } h \text{ in } G; \\
\{ g \text{ is a } \xi \text{-optimal flow in } G \} \\
\text{return } g. \\
\]

Figure 6: Tardos and Wayne’s Push-Relabel algorithm.
\begin{algorithm}
\textbf{INPUT:} a generalised flow network $G$
\begin{algorithmic}
\State $f \leftarrow \text{zero flow in } G$;
\State $k \leftarrow 0$;
\Loop
\State $k \leftarrow k + 1$;
\State cancel all flow generating cycles in $G_f$ and compute the canonical labeling $\mu$;
\If{$\text{TotResEx} \geq \xi e_f(t)$} \textbf{terminate};
\State $h \leftarrow \text{PHASE}(\xi_k)$;
\State $f \leftarrow f + h$;
\EndLoop
\EndAlgorithm
\textbf{OUTPUT:} $\xi$-optimal flow $f$.
\end{algorithmic}
\end{algorithm}

PHASE($\xi$):
\begin{algorithmic}
\State $G_f$ does not have flow generating cycles, $\mu$ is the canonical labeling in $G_f$
\State $b \leftarrow (1 + \xi)^{1/n}$;
\State let $\bar{\gamma}_\mu(v, w)$ be $\gamma_\mu(v, w)$ rounded down to integer power of $b$, for each residual edge $(v, w)$ in $G_f$;
\While{$v \in G_f$ with positive excess}
\If{exists an admissible edge $(v, w)$ \{ a residual edge in $G_f$ with $\bar{\gamma}_\mu(v, w) > 1$ \} \textbf{then}}
\State \textbf{push:} update $f$ by sending \text{min}\{$e_f(v), u_f(v, w)$} units of flow along $(v, w)$;
\Else
\State \textbf{relabel:} $\mu(v) \leftarrow \mu(v)/b^{1/n}$;
\EndIf
\EndWhile
\State $[\ell_{\text{opt}}(t) - e_{f'}(t)] \leq [\xi/(1 + \xi)] \cdot [\ell_{\text{opt}}(t) - e_{f''}(t)]$
\State where $f'$ and $f''$ are the flows at the beginning and at the end of the computation.
\end{algorithmic}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{algorithm}
\caption{Adaptation of Tardos Wayne's Push-Relabel algorithm.}
\end{figure}

\{(c, q) : q = 1, 2, \ldots, Q\} form a clique. Thus a currency $c$ at a market $q$ can be exchanged into any other currency within the same market and can be transfer into the same currency at any other market. Similarly as above, generator \textsc{GridOfClques} has also parameters $\gamma_{\text{min}}, \gamma_{\text{max}}, u_{\text{min}}$ and $u_{\text{max}}$ defining ranges of the edge gains and capacities, and $\gamma_{\text{min}}$ and $\gamma_{\text{max}}$ should preferably be set close to 1.

7 Experiments

In our experiments, we consider the various values of nodes as shown in the following Table 1.

8 Conclusion

- Excess scaling: full SP computation from scratch seems to be the bottleneck, we have to come up with a suitable practically fast update of the SP tree.

- Push-Relabel:
  1. periodical computation of “good” labels may significantly improve the running times;
  2. avoiding of “canceling flow generating cycles” could be done by saturating edges with relabeled gains $> 1 + \epsilon$ (but “deficit” nodes would be created, so the overall benefit of this approach is not clear).

- Truemper’s algorithm with cost scaling seems to be an interesting option, which is worth experimental evaluation.

- A simple method which combines together the excess scaling with the gain scaling is needed.
The new approximation “packing” algorithms for generalised maximum flow problem ([?], [?]), based on [?]) may also be worth experimental evaluation.
Figure 9: Running times on inputs created by the generator LAYERSX.

References


Figure 10: Running times on inputs created by the generator GridOfClique.

<table>
<thead>
<tr>
<th>nodes/edges</th>
<th>HIGHESTPATH</th>
<th>MAXFLOW</th>
<th>EXSC0</th>
<th>EXSC1</th>
<th>EXSC2</th>
<th>EXSC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/1000</td>
<td>0.10</td>
<td>0.23</td>
<td>38.16</td>
<td>0.30</td>
<td>0.38</td>
<td>0.09</td>
</tr>
<tr>
<td>200/3000</td>
<td>0.45</td>
<td>1.05</td>
<td>341.18</td>
<td>1.93</td>
<td>1.92</td>
<td>0.36</td>
</tr>
<tr>
<td>400/8000</td>
<td>1.67</td>
<td>4.16</td>
<td>-</td>
<td>6.91</td>
<td>8.44</td>
<td>1.64</td>
</tr>
<tr>
<td>800/23000</td>
<td>9.05</td>
<td>21.84</td>
<td>-</td>
<td>32.00</td>
<td>40.93</td>
<td>7.98</td>
</tr>
</tbody>
</table>


(a)  
<table>
<thead>
<tr>
<th>nodes/edges</th>
<th>ExSc3</th>
<th>PushRelabel</th>
<th>CPLEX-p</th>
<th>CPLEX-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/500</td>
<td>0.11</td>
<td>0.06</td>
<td>0.03</td>
<td>0.04</td>
</tr>
<tr>
<td>200/1000</td>
<td>0.37</td>
<td>0.19</td>
<td>0.08</td>
<td>0.14</td>
</tr>
<tr>
<td>400/2000</td>
<td>1.13</td>
<td>0.78</td>
<td>0.22</td>
<td>0.39</td>
</tr>
<tr>
<td>800/4000</td>
<td>5.78</td>
<td>2.68</td>
<td>0.72</td>
<td>1.54</td>
</tr>
<tr>
<td>1600/8000</td>
<td>27.64</td>
<td>16.80</td>
<td>2.35</td>
<td>5.00</td>
</tr>
<tr>
<td>3200/16000</td>
<td>122.82</td>
<td>102.37</td>
<td>31.32</td>
<td>16.00</td>
</tr>
</tbody>
</table>

(b)  
<table>
<thead>
<tr>
<th>nodes/edges</th>
<th>ExSc3</th>
<th>PushRelabel</th>
<th>CPLEX-p</th>
<th>CPLEX-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/1000</td>
<td>0.08</td>
<td>0.09</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td>200/3000</td>
<td>0.21</td>
<td>0.27</td>
<td>0.11</td>
<td>0.27</td>
</tr>
<tr>
<td>400/8000</td>
<td>1.45</td>
<td>2.06</td>
<td>0.26</td>
<td>1.37</td>
</tr>
<tr>
<td>800/23000</td>
<td>7.16</td>
<td>11.95</td>
<td>1.89</td>
<td>9.73</td>
</tr>
</tbody>
</table>

Figure 11: Running times on inputs created by the generators (a) LAYERSX, and (b) GridOfCliques.

Figure 12: Average running times on inputs created by generator LAYERSX.