On Végh's Strongly Polynomial Algorithm for Generalized Flows

by

Venus Hiu Ling Lo

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2014

Abstract

This thesis contains an exposition of the new strongly polynomial algorithm for the generalized flow problem by László Végh (2013). It has been a long-standing open question whether such an algorithm exists, until it was resolved by Végh in 2013. Generalized flows have many applications in economic problems, such as transportation of goods and foreign currency exchange. The main presentation follows Végh's paper, but this exposition contains some simplifications and differences in the algorithm and its analysis. The main difference is that we consider the running time of the strongly polynomial algorithm up to one arc contraction before starting fresh on a smaller network. This increases the running time of the algorithm slightly, but the analysis becomes easier.

Acknowledgements

I would like to thank my supervisor, Professor Joseph Cheriyan for training me patiently throughout my time in this graduate program. He spent many hours in helping me prepare both this thesis and my thesis presentations. I would also like to thank Ian Post for his help. He provided many insights into the construction of the algorithm and simpler ways to understand the proofs. He also helped me develop a network-based explanation of Hochbaum's construction. Without their help in understanding the literature, this thesis would not have been possible.

I would like to thank Professors Chaitanya Swamy and Ricardo Fukasawa for being my readers and giving me invaluable feedback. I would also like to thank my officemate, Zhihan Gao, for proofreading.

Finally, I would like to thank my parents for their support, regardless of what I decide to pursue in life. I thank God for giving me this amazing opportunity to study at Waterloo.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Very recently, László Végh introduced a strongly polynomial algorithm for the generalized flow problem. The search for such an algorithm had been a long-standing open problem and his progress was considered significant progress in the network flow community.

This thesis is an exposition on Végh's algorithm and my focus will be on understanding how Végh achieved a strongly polynomial running time. Tables of the notations for describing the network and for the generalized flow problem up to Chapter 1 are provided on page 20 at the end of the chapter. This will be useful for readers who are interested in a particular section rather than the full thesis.

1.1 The Search for Strongly Polynomial Algorithms

The generalized flow problem is an extension of the traditional maximum flow problem, where arcs can modify the amount of flow passing through them. Algorithms to solve the problem have existed since research commenced around the 1960s by Dantzig and Jewell [7], but Végh's algorithm in [9] is the first strongly polynomial algorithm.

Definition 1. *An algorithm for a flow problem is strongly polynomial if its running time is a polynomial of n and m, where n is the size of the node set and m is the size of the arc set. Furthermore, if we require that the inputs are rational numbers, then all numbers that are computed during the algorithm must also be rational numbers with bit sizes that are polynomially bounded in the bit sizes of the original inputs.* [9]

This is different than a polynomial algorithm, also known as a weakly polynomial algorithm, where the running time may also be dependent on $\log U$, where U is an upper bound on the numerical inputs (e.g. capacities, costs).

There are several reasons why one might want to find a strongly polynomial algorithm for network flow problem. Firstly, strongly polynomial algorithms could perform better in practice than weakly polynomial algorithms; this is the the case for the traditional maximum flow problem. Numerical inputs, like capacities, could be very large and it may be best if the running time is independent of these inputs. But strongly polynomial algorithms for more general network flow problems may not perform better than other algorithms in practice, because most "real world" instances have restricted bit sizes (e.g. largest numeric data fits into 64 bits) [4]. In the case of the minimum cost flow problem, current weakly polynomial algorithms run faster than strongly polynomial algorithms in practice, that is, with "real world" data.

Secondly, strongly polynomial algorithms are viewed as the holy grail in the search for better algorithms for optimization problems [4]. In 1972, Edmonds and Karp originally posed the question of finding a strongly polynomial algorithm for the minimum cost flow problem, Éva Tardos gave the first such algorithm in 1985. For the generalized flow problem, the first weakly polynomial algorithm was introduced in 1991 by Goldberg, Plotkin, and Tardos, but there was little progress towards strongly polynomial algorithms until Végh's contribution.

The final, and perhaps most important reason, is the relationship between generalized flow problems and linear programming (LP) [9]. The generalized flow problem is closely related to another extension, called the minimum cost generalized flow problem (MCGF). These two problems are related in the same way that the traditional maximum flow problem and minimum cost flow problem are related. It turns out that MCGF is also closely related to general linear programming. A general linear program can be transformed to another linear program with at most three non-zero entries per column. On the other hand, Hochbaum proved that any linear program with at most two non-zero entries per column can be transformed to an instance of MCGF (see Section 1.7). Being able to solve MCGF in strongly polynomial time is a step towards solving any linear program in strongly polynomial time, another long-standing open problem. On the other hand, LPs having three non-zero entries per column could be much harder to solve than LPs having at most two-non-zero entries per column (recall that 2-SAT is solvable in polynomial time but not 3-SAT). In this case, we would be interested in the largest class of linear programs that can be solved in strongly polynomial running time.

The next table summarizes the progress for finding strongly polynomial algorithms for network flow problems.

Problem	Runtime	Year
Max flow	$O(n^2m)$	Dinic 1970
Min cost flow	$O(nm^{3/2})\times$ runtime of max flow	Tardos 1985
Generalized flow	$O(n^3m^2)$	Végh 2013
MCGF (≤ 2 non-zero per col)	-	-
Linear programming (≤ 3 non-zero per col)	-	-

Table 1.1: History of strongly polynomial algorithms for network flow problems

1.2 Problem Setup

In the generalized flow problem, we are given a network G with node set V and arc set E. Let $n = |V|$ and $m = |E|$. Throughout this thesis, all paths and cycles refer to directed paths and cycles.

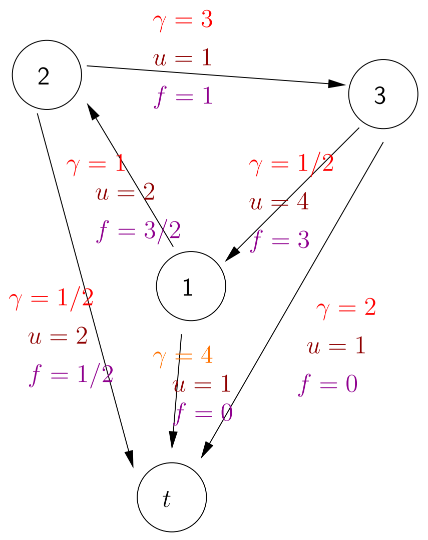


Figure 1.1: Example of a generalized flow

Similarly to the traditional maximum flow problem, we are trying to maximize the flow sent to a special sink node t subject to the capacities $u : E \to \mathbb{R}$ on the arcs (see Figure 1.1). A feasible flow is any flow $f : E \to \mathbb{R}_{\geq 0}$ that has non-negative net flow (i.e. total inflow less total outflow) at every node other than t, and moreover, satisfies capacity constraints. The first difference is that we also have gain factors, $\gamma : E \to \mathbb{R}_{>0}$, on the arcs. When ϵ units of flow leave node i and go into arc ij, the flow is multiplied by the gain factor γ_{ij} so that $\gamma_{ij}\epsilon$ units of flow arrive at node j. If $\gamma_{ij} > 1$, then we generate flow over the arc ij. In contrast, if $\gamma_{ij} < 1$, then we lose flow. Note that the arc capacities apply to the outgoing flow, so that u_{ij} bounds the flow leaving node i. That is, the capacity constraint for any arc ij is $f_{ij} \leq u_{ij}$ (and not $\gamma_{ij} f_{ij} \leq u_{ij}$). By a positive flow, we mean a nonzero, feasible flow.

The second difference is that the network does not have a source node to provide flow. Instead, the network is able to generate its own positive flow if it contains a cycle such that the product of the gain factors of its arcs is greater than 1. In this case, we can send some $\epsilon > 0$ units of flow around the cycle and end with more than ϵ units of flow arriving at the start node of the cycle. The excess created can be sent to the sink t. It is easy to observe that a positive flow exists only if such a cycle exists.

The traditional maximum flow problem is a special case of the generalized flow problem. To construct the maximum flow problem, we can use the same network and give all arcs a gain factor of 1. In order to allow for an arbitrary amount of flow to leave the source s, we can construct a loop at s with a gain factor greater than 1 and infinite capacity. By pushing flow on this loop, we can create any amount of flow at s to be sent over to t, capturing the maximum flow problem.

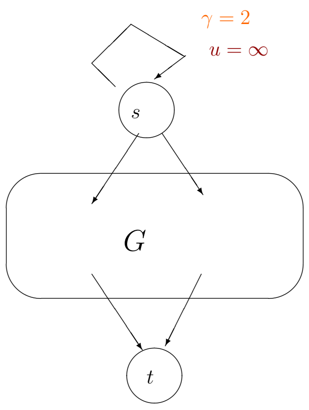


Figure 1.2: Using generalized flow to represent traditional maximum flow. All arcs other than the loop at s gets $\gamma_{ij} = 1$.

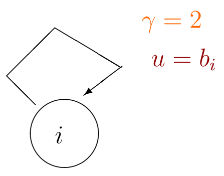
1.3 Linear Program and Transformations

Let f_{ij} denote the flow leaving node i on arc ij and f_{ji} denote the flow entering node i on arc ji. It is possible to have both arcs ij and ji in our network. The linear program to represent the generalized flow program is:

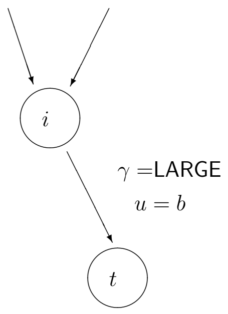
$$\begin{aligned} \max \sum_{jt \in E} \gamma_{jt} f_{jt} - \sum_{tj \in E} f_{tj} & \\ \sum_{ji \in E} \gamma_{ji} f_{ji} - \sum_{ij \in E} f_{ij} \geq 0 & \qquad \forall i \in V - t \\ 0 \leq f_{ij} \leq u_{ij} & \qquad \forall ij \in E \end{aligned}$$

Some of the literature uses an equation constraint for the first constraint. However, this relaxation has the same optimal solution [7], because we can always remove extra excess by pushing flow in reverse on the cycle that created the excess.

Using Capacitated Arcs to Represent Supplies and Demands
In the original setup, nodes do not have supplies and demands, but we can simulate supplies and demands using gain factors and capacities on arcs. If a node has a supply of b units, we can set up a loop with b units of capacity and gain factor of 2 (Figure 1.3a). Pushing b units along this loop will generate b units of excess. On the other hand, if a node i has a demand of b units, we can set up an arc it with capacity b and a very large gain factor, say $(\max_{ij \in E}\{\gamma_{ij}, 1\})^m$ (Figure 1.3b). Then every optimal solution must use all b units of capacity as no other path can generate as much flow into t. If the capacity is not used, there is no feasible solution. By removing the arc it subsequently, node i must satisfy its demand.



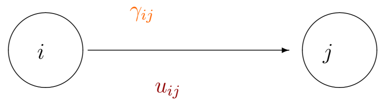
(a) Supply node



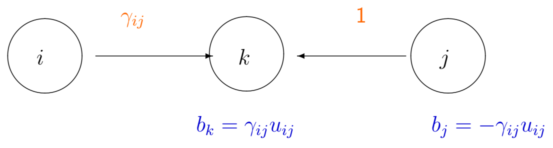
(b) Demand node

Figure 1.3: Transformation of supply and demand nodes to capacitated arcs

Using Supplies and Demands to Replace Capacitated Arcs
We can also reverse the above process and use supply and demand nodes to change a capacitated network to an uncapacitated network [9]. We can replace an arc ij with a new node k, and add arcs ik and jk with infinite capacities. The gain factors on these arcs are $\gamma'_{ik} = \gamma_{ij}$ and $\gamma'_{jk} = 1$ (Figure 1.4). Let a positive value represent a demand and negative value represent a supply. Node k is given a demand of $\gamma_{ij}u_{ij}$ and node j is given a supply of $-\gamma_{ij}u_{ij}$. Sending some $f_{ij} \leq u_{ij}$ units from i to j is equivalent to sending f_{ij} units from i to k, and having the remainder of the demand satisfied by sending $\gamma_{ij}u_{ij} - \gamma_{ij}f_{ij}$ units from j to k. This results in $\min\{\gamma_{ij}f_{ij}, \gamma_{ij}u_{ij}\}$ units at node j. This is the same as sending f_{ij} units on arc ij in the capacitated network.



(a) Capacitated arc



(b) Uncapacitated arc with supplies and demands

Figure 1.4: Transformation of capacitated arc to two uncapacitated arcs

1.4 Applications and Extensions

The generalized flow problem has many important applications, particularly in economical problems.

Transportation of Goods with Losses: Consider the problem of sending goods from locations to a centralized processing plant, such as sending oil from different wells to a refinery plant. During transportation, there may be losses due to inefficiency and evaporation inside the pipes. Clearly, we want to minimize our wastage by maximizing the amount of oil reaching the plant. We can model the pipelines with arcs and wells with nodes, where the sink represents our plant. The gain factors on the arcs are set to be less than 1, to model the retention rate after travelling through a particular pipe.

Foreign Currency Exchange Market and Arbitrage [1]: Let the nodes represent the different currencies with the sink t being the local currency. If clients can exchange currency i for currency j at the bank, then we will add an arc ij with γ_{ij} equal to the exchange rate. Arbitrage occurs when we can make a profit by a sequence of currency exchanges. In the foreign currency exchange scenario, this would happen if we can borrow one unit of currency i, exchange into different currencies in a cycle and end up with more than one unit of currency i. We could then repay the debt and exchange the profit to the local currency. The problem of finding an arbitrage opportunity is equivalent to finding a positive flow in the generalized flow network.

Extensions to Other Problems: If we replace the objective function with an arbitrary linear objective function $\min \sum_{ij} c_{ij} f_{ij}$, then this becomes a minimum cost generalized flow problem (MCGF). In this case, we would not have a sink node to maximize flow into, but rather demands at a subset of nodes which we must satisfy. Some of the applications that can be modeled with this extension includes the oil transportation problem above, except that we want to minimize the cost of sending a fixed amount of flow.

1.5 Optimality Conditions for the Standard Model

We want to understand the optimality conditions for the standard generalized flow model first, because early algorithms were based on this model. In Section 2.1, I will give optimality conditions that are specialized for Végh's setup.

1.5.1 Duality and Complementary Slackness Conditions

The dual linear program, with a dual variable z_i for each node $i \in V - t$ and a dual variable w_{ij} for each arc $ij \in E$, is as follows:

$$\begin{aligned}
&\min \sum_{ij \in E} u_{ij} w_{ij} && \\
&z_i - \gamma_{ij} z_j + w_{ij} \geq 0 && \forall ij \in E; i, j \neq t \\
&z_i + w_{it} \geq \gamma_{it} && \forall it \in E \\
&-\gamma_{ti} z_i + w_{ti} \geq -1 && \forall ti \in E \\
&z \geq 0, w \geq 0 &&
\end{aligned}$$

It is customary, however, to use the inverse of a dual variable z_i when we discuss generalized flows: $\mu_i = 1/z_i$. The transformed dual program is:

$$\begin{aligned}
&\min \sum_{i \in V-t} u_{ij} w_{ij} && \\
&\gamma_{ij} \frac{\mu_i}{\mu_j} - w_{ij}\mu_i \leq 1 && \forall ij \in E \\
&\mu_i > 0 && \forall i \in V - t \\
&\mu_t = 1 &&
\end{aligned}$$

It is possible for $z_i = 0$ to occur, which means that $1/\mu_i = 0$ and $\mu_i = \infty$. We will use the convention that $\infty/\infty = 0$ so that the label is feasible when the associated dual solution is feasible.

We will use the dual program to identify the complementary slackness conditions. Let the excess be the net flow of a node:

$$e_i = \sum_{j:ji \in E} \gamma_{ji} f_{ji} - \sum_{j:ij \in E} f_{ij} \qquad \forall i \in V$$

The complementary slackness conditions for the standard form of the generalized flow are:

- $e_i = 0$ or $1/\mu_i = 0$ (i.e. $\mu_i = \infty$), $\forall i \in V - t$
- $f_{ij} = u_{ij}$ or $w_{ij} = 0$, $\forall ij \in E$
- $f_{ij} = 0$ or $\gamma_{ij} \frac{\mu_i}{\mu_j} - w_{ij}\mu_i = 1$, $\forall ij \in E$

It is easier to interpret the second and third conditions in terms of the residual graph. The residual graph G_f (ignoring the numerical values of the residual capacities) is defined in the same way as in the traditional maximum flow. The node set is $V(G)$ and the arc set is $E_f = \{ij : ij \in E, f_{ij} < u_{ij}\} \cup \{ji : ij \in E, f_{ij} > 0\}$. For a reverse arc ji, we let $\gamma_{ji} = 1/\gamma_{ij}$. It is important to note that we can have both an original arc ji and a reverse arc ji, and that they can have different gain factors. When I mention ji throughout this thesis, it will be clear from the context whether I am referring to an original arc or a reverse arc in the residual graph.

Let us focus on the last two complementary slackness conditions. If $f_{ij} < u_{ij}$, then the second complementary slackness condition and the dual constraint implies that $\gamma_{ij} \frac{\mu_i}{\mu_j} \leq 1$.

If $f_{ij} > 0$, then the third complementary slackness condition and $\gamma_{ji} = 1/\gamma_{ij}$ rearranges to $\gamma_{ji}\frac{\mu_j}{\mu_i} = \frac{1}{1+w_{ij}\mu_i} \leq 1$. The last inequality follows from $w_{ij} \geq 0$ and $\mu_i > 0$. We can simplify this to requiring that $\gamma_{ij}\frac{\mu_i}{\mu_j} \leq 1$ for all ij in the residual graph. This is summarized in the following definition:

Definition 2. *Let μ be feasible labels. Then μ is a set of conservative labels for f if $\gamma_{ij}\frac{\mu_i}{\mu_j} \leq 1$ for all ij in the residual graph of f.*

The optimality conditions simplify to having a pair of feasible primal and dual solutions (f, μ) such that:

- $e_i = 0$ or $\mu_i = \infty$, $\forall i \in V - t$
- μ is a set of conservative labels.

We already saw that if (f, μ, w) satisfy the complementary slackness conditions, then (f, μ) satisfy the new optimality conditions. On the other hand, if we have (f, μ) such that the above conditions are satisfied, then it is easy to find w so that (f, μ, w) satisfy the complementary slackness conditions. Consider any arc ij. If $f_{ij} < u_{ij}$, then set $w_{ij} = 0$. If $f_{ij} = u_{ij} > 0$, then set w_{ij} by rearranging the condition $\gamma_{ij}\frac{\mu_i}{\mu_j} - w_{ij}\mu_i = 1$:

$$w_{ij} = \frac{1 - \gamma_{ij}\frac{\mu_i}{\mu_j}}{-\mu_i} = \frac{\gamma_{ij}}{\mu_j} - \frac{1}{\mu_i} \geq 0$$

The last inequality is true because the reverse arc ji is in E_f and μ is a set of conservative labels. This means that $\gamma_{ji}\frac{\mu_j}{\mu_i} \leq 1$, which rearranges to $\frac{1}{\mu_i} \leq \frac{1}{\gamma_{ji}\mu_j} = \frac{\gamma_{ij}}{\mu_j}$.

Not only are labels important in certifying that a solution is optimal, they can help us identify the best path to send excess to t.

Definition 3. *Given conservative labels μ, an arc ij is tight if $\gamma_{ij}\frac{\mu_i}{\mu_j} = 1$, and a path P is tight if all arcs $ij \in P$ are tight.*

Let k be a node such that $e_k > 0$, and P be any path from k to t on the residual graph. If P is tight for conservative labels μ, then it is easy to show that P is the path of highest gain from k to t. Consider the products of the $\gamma_{ij}\frac{\mu_i}{\mu_j}$ for all ij on path P. Using the fact that $\mu_t = 1$ for feasible labels μ, we have:

$$\mu_k \prod_{ij \in P} \gamma_{ij} = \prod_{ij \in P} \gamma_{ij}\frac{\mu_i}{\mu_j} \leq 1$$

The equality is due to $\mu_t = 1$ in a feasible label. The above inequality implies that if we send one unit of flow from k to t, we can get at most $1/\mu_k$ units arriving at t, with equality holding if we use a tight path. If there is no tight path from k to t, then we need to update the labels until we can find a tight path without violating dual feasibility.

1.5.2 Optimality Conditions via Flow-Generating Cycles

Another method to certify an optimal flow is through finding flow-generating cycles.

Definition 4. *A cycle C is a flow-generating cycle if $\prod_{ij \in C} \gamma_{ij} > 1$.*

If we have a flow-generating cycle C, then we can push ϵ amount of flow starting at some node $i \in C$, and receive more than ϵ flow arriving back at i, thereby creating excess at i.

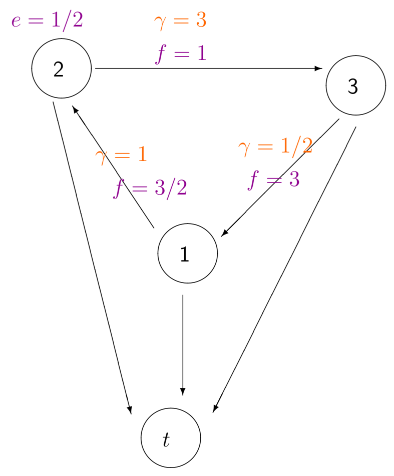


Figure 1.5: $C = 2, 3, 1$ is a flow-generating cycle. If we start by pushing one unit from node 2 around the cycle, 1.5 units arrive back at node 2, creating an excess of 0.5 units.

If we can find a flow-generating cycle C in G_f, and a path from node $i \in C$ to the sink t, then we can increase the flow to t. Therefore, another way to state the optimality conditions is that there is no i-t path in G_f for node $i \in V - t$ such that [7]:

- i is on a flow-generating cycle in G_f; OR

- $e_i > 0$.

Let us look at the relationship between flow-generating cycles and labels. Conservative labels can only exist for f if G_f contains no flow-generating cycles. For a cycle C contained in G_f, the requirement $\gamma_{ij}\frac{\mu_i}{\mu_j} \leq 1$ for arcs $ij \in C$ implies that $\prod_{ij\in C}\gamma_{ij} = \prod_{ij\in C}\gamma_{ij}\frac{\mu_i}{\mu_j} \leq 1$.

1.6 Earlier Algorithms for Generalized Flows

The generalized flow problem has been studied extensively since 1960s by Dantzig and Jewell. This section looks at how early algorithms solved the generalized flow problem.

Algorithms generally find an initial feasible flow by cancelling all flow-generating cycles in the residual graph. Cancelling flow-generating cycles simply means that we find a flow-generating cycle in G_f and push flow around it until an arc uses up its residual capacity. We can repeat this procedure until there are no more flow-generating cycles. The issue is that we could create new flow-generating cycles when we cancel an old one. This can be solved by always cancelling the cycle with the maximum mean-gain $\gamma(C)^{1/|C|}$ [7]. We can run Radzik's maximum-mean-gain-cycle-cancelling algorithm to find an initial flow with no flow-generating cycles in the residual graph in $O(m^2 n \log^2 n)$ runtime [6, 7]. Subsequently, we can try to send the excesses created to t using one of the algorithms described below.

1.6.1 Onaga's Algorithm Based on Augmenting Paths

Onaga's algorithm is built on the augmenting path algorithm of the traditional maximum flow problem [5]. First we choose a node k with $e_k > 0$. Then we send the excess to t along a tight path up to the maximum residual capacity. We repeat this process until all nodes with positive excesses do not have a path to t in G_f. The labels may need to be updated from time to time to tighten paths. The correctness of the algorithm is based on the next lemma.

Lemma 5. *Assume we have conservative labels μ. If $e_k > 0$ and we send flow from k to t along a tight path, then no new flow-generating cycles are created in G_f.*

Proof. Assume by contradiction that we introduce a new arc ij into G_f by pushing flow on the reverse arc ji, such that ij is part of a new flow-generating cycle C. Since ji was tight, $\gamma_{ji}\frac{\mu_j}{\mu_i} = 1 = \gamma_{ij}\frac{\mu_i}{\mu_j}$, so ij is also tight. This means that there is some other arc $pq \in (C - ij)$ such that $\gamma_{pq}\frac{\mu_p}{\mu_q} > 1$. This violates the assumption that μ is a conservative label. □

Since this algorithm was built on the augment path algorithm for traditional maximum flow, this algorithm has pseudo-polynomial worst-case runtime.

1.6.2 Goldberg-Plotkin-Tardos Fat-path Algorithm

The fat-path algorithm was the first scaling algorithm introduced for generalized flows [2]. Let $\Delta \in \mathbb{R}_{>0}$ be our scaling parameter. A Δ-phase refers to the set of iterations with scaling parameter Δ. Whenever we augment flow in the Δ-phase, we require that Δ units of flow reaches t. We still want to use tight paths to send flow in order to ensure that we do not create new flow-generating cycles, but now we also need to ensure that the paths we use have sufficient residual capacity.

Recall from the end of Subsection 1.5.1 that if $e_k > 0$ and G_f contains a k-t path, then one unit of flow leaving k will result in at most $1/\mu_k$ units of flow arriving at t. This means that $\Delta\mu_k$ units must leave node k in order for Δ units to arrive at t. Furthermore, if we sent flow along a path P, then every $ij \in P$ needs to have residual capacity of $\Delta\mu_i$ to accommodate Δ units arriving at t. The algorithm runs as follows after flow-generating cycles are cancelled:

1. Start with some $\Delta = \Delta^{\text{start}}$ and $f = 0$.
2. Find conservative labels μ, with the additional requirement that there is a tight i-t path for all $i \in V - t$, if such a path exists in G_f.
3. In phase Δ:
    - Build a subgraph H of the residual graph with all the original nodes. Arc ij is in H if $ij \in E_f$ and its residual capacity in G_f is at least $\Delta\mu_i$.
    - Set μ to be conservative labels for H, with the additional requirement that there is a tight i-t path for all $i \in V - t$, if such a path exists in H.
    - While there exists a node $k \in V - t$ such that $e_k \geq \Delta\mu_k$ and our subgraph contains a tight k-t path P_k, send $\Delta\mu_k$ units from k along P_k so that Δ units arrive at t.
4. Set $\Delta = \Delta/2$, and cancel new flow-generating cycles. If $\Delta > \Delta^{\text{end}}$, then go back to step 3.

The last step requires us to cancel new flow-generating cycles, because we are not necessarily using a highest gain path but rather a highest gain path with respect to the arcs with sufficient residual capacity.

The algorithm terminates when we reach a small $\Delta = \Delta^{\text{end}}$. When our scaling parameter Δ^{end} is small, we will obtain optimal labels [7]. Optimal labels can help us find an optimal flow with one traditional maximum flow computation. This will be shown in details in Section 3.4 when I present the termination procedure for Végh's algorithm.

Both Δ^{start} and Δ^{end} are chosen to be polynomials of the network size and logarithmic in the numerical input. Since we reduce Δ by a factor of 2 at the end of each phase, it is clear that this algorithm is inherently weakly polynomial due to the number of phases.

1.7 Hochbaum's Transformation to Minimum Cost Generalized Flows

In Section 1.1, I mentioned that every linear program with at most two non-zero entries per column can be transformed to a MCGF problem in polynomial time [3]. Before I show the construction, I will first give a pictorial interpretation to motivate the construction.

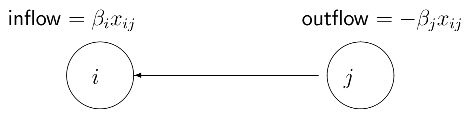
Let our current linear program, (LP-1), be in the following form:

$$\begin{aligned} \max\ c^T x \qquad\qquad & \text{(LP-1)} \\ Ax = b \qquad\qquad & \\ x \geq 0 \qquad\qquad & \end{aligned}$$

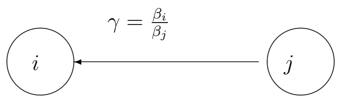
In the linear program of a generalized flow problem, we have a positive entry and a [-1] in a column representing an arc. In comparison, (LP-1) could have any two non-zero entries in a column, or even one non-zero entry. If a column of matrix A has non-zero entries in rows i and j, we will call the corresponding variable x_{ij} (i.e. column ij). It is possible to have a column corresponding to x_{ii} if there is only one non-zero entry in row i. We can interpret the rows as nodes, and the columns as arcs that are directed in an awkward manner described in the next subsection. Thus we are not concerned with the ordering of ij versus ji yet. The b_i on the right side of (LP-1) represents the demand or supply at a node i.

1.7.1 Intuition Behind Transformation: 2 Non-zero Entries

(LP-1) could have a column ij with both a positive entry β_i and a negative entry $-\beta_j$, where $\beta_i, \beta_j > 0$. This looks like an arc ij, except that the outflow to j is $-\beta_j x_{ij}$ and the inflow to i is $\beta_i x_{ij}$ when we compute the excesses at the two nodes (Figure 1.6a). However, we can scale the variable x_{ij} by β_j to arrive at an arc with a gain factor of β_i/β_j (Figure 1.6b). A flow f_{ij} would translate to a solution of (LP-1) by setting $x_{ij} = \beta_j f_{ij}$.



(a) Before scaling



(b) After scaling; $x_{ij} = \beta_j f_{ij}$

Figure 1.6: Column with one positive and one negative entry.

It becomes difficult when column ij has two positive entries or two negative entries. Let us focus on the positive case with β_i and β_j; the negative case is similar. Pictorially, this suggests a two-headed arc $\overleftrightarrow{ij}$, with inflows of $\beta_i x_{ij}$ and $\beta_j x_{ij}$ at nodes i and j respectively.

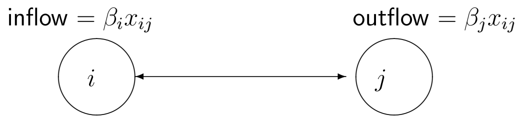


Figure 1.7: Column with two positive entries.

We may be tempted to multiply row j by -1, but this could affect another column that represents a normal arc incident to j. Instead, we will attach a new graph G' to G, where G' and $j' \in V'$ are mirror copies of G and $j \in V$ respectively. We give j' the negative supply/demand $-b_j$. For any arc originally with a head at j, we will have an arc with tail at j', and vice versa. This means we can replace the original double-headed ij with an arc $j'i$ to represent the flow into i. On $j'i$, the outflow at j' is multiplied by $-\beta_j$ and the inflow at i is multiplied by β_i. Similarly, we should create a new node i' to act as the mirror copy of i, and an arc $i'j$ to replace the inflow into j (Figure 1.8).

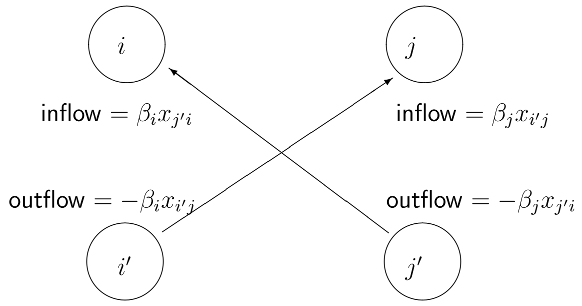

i
j
inflow $= \beta_i x_{j'i}$
inflow $= \beta_j x_{i'j}$
outflow $= -\beta_i x_{i'j}$
outflow $= -\beta_j x_{j'i}$
i'
j'

Figure 1.8: Replacing a two-headed arc with two arcs and two new auxiliary nodes.

We can scale the flow in the same manner as Figure 1.6. Furthermore, we can do the same transformation for a two-tailed node, by creating new arcs ij' and ji'.

Finally, in order to include the reverse arc of a normal arc ij into G', we can add an arc $j'i'$.

1.7.2 Intuition Behind Transformation: 1 Non-zero Entry

We can think of these columns as loops. First assume that the entry is $\beta_i > 0$ at column ii. This can be be viewed as a loop with gain factor $\beta_i + 1$, so that 1 unit leaving i will come back as $\beta_i + 1$ units, giving us a net flow of β_i. On the mirror node i' in G', we want a loop to result in a net flow of $-\beta_i = (1 - \beta_i) - 1$. This means adding an arc with gain factor $1 - \beta_i$. However, a generalized flow requires that all gain factors are positive. To ensure that $\beta_i + 1 > 0$ and $1 - \beta_i > 0$, we should scale to get $1 > \beta_i > -1$. The above construction is exactly the same if the original entry is $-\beta_i$.

1.7.3 Hochbaum's Construction

Our new linear program (LP-2) will be built as follows [3]:

1. The rows will be $V \cup V'$ where V' is a second copy of the initial rows. Let $b_{i'} = -b_i$ for row i'.

2. If column ij has two positive entries, we will replace it with two columns $i'j$ and $j'i$. Fill in the columns using Table 1.2.

3. If column ij has two negative entries, we will replace it with columns ij' and ji'. Fill in the columns using Table 1.2.

4. If column ij has a negative entry at i and a positive entry at j, we will replace it with columns ij and $j'i'$. Fill in the columns using Table 1.2.

5. If there is one non-zero entry in column ii, we will replace it with columns ii and $i'i'$. Fill in the columns using Table 1.2.

6. Set the costs on the new arcs to be 1/2 of the cost on the original arcs (i.e. Do not negate costs).

Position	Two-headed ij β_i, β_j		Two-tailed ij $-\beta_i$, $-\beta_j$		Normal ij $-\beta_i$, β_j		Loop β_i	
	$i'j$	$j'i$	ij'	ji'	ij	$j'i'$	ii	$i'i'$
i	0	β_i	$-\beta_i$	0	$-\beta_i$	0	β_i	0
j	β_j	0	0	$-\beta_j$	β_j	0	0	0
i'	$-\beta_i$	0	0	β_i	0	β_i	0	$-\beta_i$
j'	0	$-\beta_j$	β_j	0	0	$-\beta_j$	0	0

Table 1.2: This table summarizes the new columns after Hochbaum's transformation. All other entries are 0. If we negate the above inputs for the normal arc and the loop, the resulting columns would also be negated.

For simplicity, let the original arc ij be denoted e and the two new arcs be denoted $e_1 \in E_1$ and $e_2 \in E_2$ in the following lemmas and proofs. In the matrix A, we will denote the entry at row i and column e as $a_{i,e}$.

Lemma 6. *Given a feasible solution $x^{(1)}$ to (LP-1), setting $x_{e_1} = x_{e_2} = x^{(1)}_e$ will give us a feasible solution to (LP-2). Given a feasible solution $x^{(2)}$ to (LP-2), setting $\bar{x}_e = (x^{(2)}_{e_1} + x^{(2)}_{e_2})/2$ will give us a feasible solution to (LP-1). Furthermore, the objective values are preserved.*

Proof. Let us look at a fixed $i \in V$. Without loss of generality, we may assume that columns in E_1 contains the original entry at position i and columns in E_2 contains the negated entry at position i'. That is:

- At $i \in V$, $a_{i,e_1} = a_{i,e}$ and $a_{i,e_2} = 0$ for $e_1 \in E_1$ and $e_2 \in E_2$.
- At $i' \in V'$, $a_{i',e_1} = 0$ and $a_{i',e_2} = -a_{i,e}$ for $e_1 \in E_1$ and $e_2 \in E_2$.

For solution x in row i of (LP-2):

$$\sum_{e \in E_1} a_{i,e} x_e + \sum_{e \in E_2} a_{i,e} x_e = \sum_{e \in E_1} a_{i,e} x_e + 0 = b_i$$

For solution x in row i' of (LP-2):

$$\sum_{e \in E_1} a_{i',e} x_e + \sum_{e \in E_2} a_{i',e} x_e = \sum_{e \in E_2} -a_{i,e} x_e + 0 = -b_i$$

Thus, x is feasible to (LP-2). The cost of x in (LP-2) is equal to the cost of $x^{(1)}$ in (LP-1):

$$\sum_{e_1 \in E_1} \frac{1}{2} c_e x_{e_1} + \sum_{e_2 \in E_2} \frac{1}{2} c_e x_{e_2} = \sum_{e \in E} c_e x_e$$

Next, consider the feasibility of $\bar{x}$ in (LP-1). For $i \in V$:

$$\begin{aligned}\sum_{e \in E} a_{i,e} \bar{x}_e &= \sum_{e \in E} a_{i,e} \left(\frac{x^{(2)}_{e_1} + x^{(2)}_{e_2}}{2} \right) \\ &= \sum_{e_1 \in E_1} \frac{a_{i,e_1} x^{(2)}_{e_1}}{2} - \sum_{e_2 \in E_1} \frac{-a_{i,e_2} x^{(2)}_{e_2}}{2} = \frac{b_i}{2} - \frac{-b_i}{2} = b_i\end{aligned}$$

Thus $\bar{x}$ is feasible to (LP-1). The cost of $\bar{x}$ in (LP-1) is equal to the cost of $x^{(2)}$ in (LP-2):

$$\sum_{e \in E} c_e \left(\frac{x^{(2)}_{e_1} + x^{(2)}_{e_2}}{2} \right) = \sum_{e \in E_1} \frac{1}{2} c_e x^{(2)}_{e_1} + \sum_{e \in E_2} \frac{1}{2} c_e x^{(2)}_{e_1}$$

□

Finally, we should scale (LP-2) so that it is the linear program of a generalized flow problem. That is, if a column has both a positive and a negative entry, scale the negative entry to -1. If a column has only one non-zero entry, we should scale the entry so that it is in the interval $(-1, 0)$ when it is negative and in the interval $(0, 1)$ when it is positive.

1.8 Contributions of Végh's Paper

Végh's algorithm is a scaling algorithm. He achieves a strongly polynomial algorithm by introducing several new ideas and techniques. Two of the ideas, which are easier to explain on a standalone basis, are described here. Namely, he uses abundant arcs and continuous scaling. More details will be presented throughout this thesis, with a final summary at the Conclusion.

Abundant arcs have been used in previous strongly polynomial, scaling-type algorithms for minimum cost flows and maximum flows. When we are at a scaling phase Δ, we can bound the difference between the optimal solution and the current solution. Thus, arcs that have a large flow relative to Δ can be guaranteed to have positive flow in some optimal solution. Later, I will show that such an arc can be contracted so that we may work on a smaller network. As long as we can find a contraction within a strongly polynomial number of iterations, we will be able obtain a strongly polynomial algorithm. This technique will be deferred until Chapter 4.

The second technique, continuous scaling, is a new idea not used in any previous algorithm for the generalized flow problem. Historically, scaling algorithms move from one phase to the next by dividing the scaling parameter Δ by the same constant in each phase. In the Fat-Path algorithm, discussed in Subsection 1.6.2, the constant was 2. In addition to having weakly polynomial running time, another problem with scaling by a fixed constant is that we could create new flow-generating cycles. Instead, Végh's continuous scaling technique allows the algorithm to choose the best scaling parameter that will maintain certain desirable properties. The choice of the new parameter will also play a role in ensuring that abundant arcs appear within a reasonable number of iterations. The risk is that Δ could decrease quite slowly. In Section 5.1, we will see that the continuous scaling technique runs sufficiently quickly for contractions to occur within a number of iterations that is polynomial in n.

1.9 Summary

In Chapter 1, I introduced the generalized flow problem in its standard form, along with methods for identifying an optimal flow using both flow-generating cycles and labels. A flow-generating cycle is a cycle where the product of the gain factors is greater than 1, so that we can push flow around it and create an excess at the starting node. Clearly, if there is a path from a flow-generating cycle to the sink t, then we can create excess on the cycle and increase the flow into t by pushing the excess on the path. Apart from identifying optimal solutions, labels are important because they help us identify paths of highest gain to t. When we have a tight path relative to a conservative label μ, we can push flow on this path without creating new flow-generating cycles, while ensuring that we are achieving the best gain possible.

Finally, I presented two old algorithms for the generalized flow problem. The first algorithm, by Onaga, is based on a modification of the augmenting path problem. The second, by Goldberg-Plotkin-Tardos, is a scaling type algorithm. Végh's algorithm is an enhancement of a scaling algorithm with several new techniques, of which two were briefly mentioned: continuous scaling and contraction of abundant arcs.

The following table summarizes the notation that we use:

Item	Notations or Definitions
Nodes V	Lower-case letters, e.g., i, j. May be represented by numbers in some diagrams.
Arcs E	ij where i is the tail and j is the head
Set of nodes	Upper-case letters, e.g., S, T_0, T, L
Arcs in $S \subseteq V$	$E[S] = \{ij \in E : i, j \in S\}$
Arcs across cut	$E[S, V \backslash S] = \{ij \in E : i \in S, j \notin S\}$ where $S \subseteq V$

Table 1.3: Graph and Network Notations

The next table summarizes the notations that will be used for the generalized flow problem throughout this thesis, and will be expanded after each chapter.

Notation	Meaning
Chapter 1: Introduction	
γ_{ij}	Gain factor of arc ij, always > 0
b_i	Demands (> 0) or supplies (< 0) at node i
e_i	Excess at node i, $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$
f_{ij}	Flow on arc ij
μ_i	Label at node i; inverse of dual solution, always > 0
E_f	The set of residual arcs: $\{ij : f_{ij} < u_{ij}\} \cup \{ji : f_{ij} > 0\}$
G_f	The residual graph with node set V and arc set E_f

Table 1.4: Summary of Notations in Chapter 1

Chapter 2

Definitions and Notations

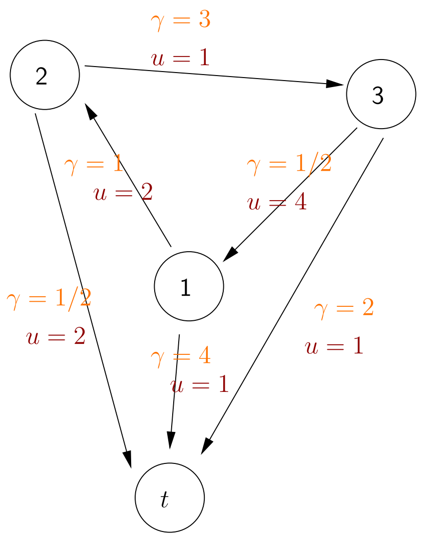
This chapter will focus on the setup and definitions in Végh's paper. A table of all the notations up to Chapter 2 is provided on page 31 at the end of the chapter. This will be useful for readers who are interested in a particular section rather than the full thesis. Some concepts were already introduced in Chapter 1, but will be repeated here for completeness.

2.1 Using an Uncapacitated Network

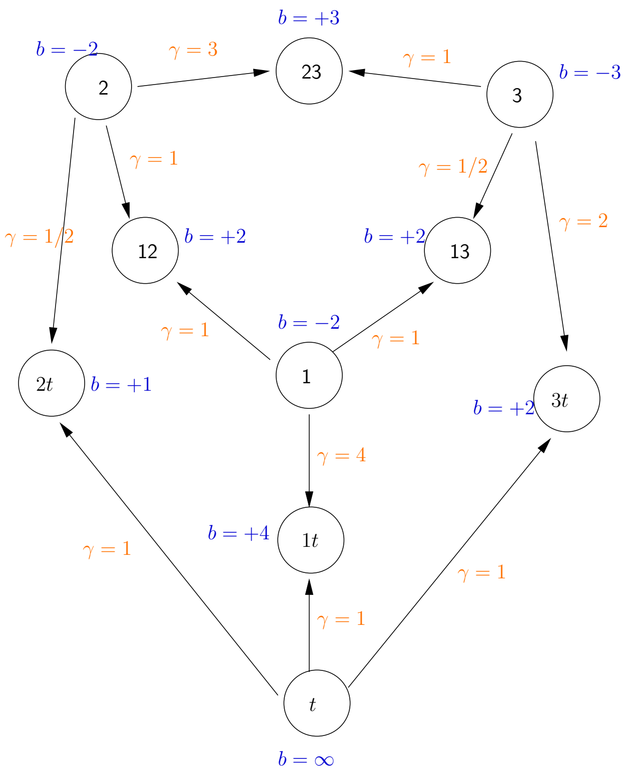
We will transform the network to its uncapacitated version with supplies and demands (see Figure 1.4). We can achieve this by adding an arc it for all $i \neq t$. As a result, we can add an auxiliary arc it, with very small gain factor like $\gamma_{it} = 1/U$ for large U (see Chapter 6 to define an appropriate U). The arc it is only used if there is no other path to send excess to t, and left out of all the figures to keep them clean. If we must use it, that means $e_i > 0$ in the optimal solution when we translate back to the original, capacitated network. Our example from Figure 1.1 transforms into Figure 2.1.

The primal and dual linear programs of the uncapacitated generalized flow problem are:

$$\begin{aligned}
&\max \sum_{jt\in E} \gamma_{jt} f_{jt} - \sum_{tj\in E} f_{tj} \\
&\sum_{j:ji\in E} \gamma_{ji} f_{ji} - \sum_{j:ij\in E} f_{ij} \geq b_i \quad \forall i \in V - t \\
&f \geq 0
\end{aligned}
\qquad
\begin{aligned}
&\min \sum_{i\in V-t} -b_i z_i \\
&z_i - \gamma_{ij} z_j \geq 0 && \forall ij \in E; i,j \neq t \\
&z_i \geq \gamma_{it} && \forall it \in E \\
&-\gamma_{ti} z_i \geq -1 && \forall ti \in E \\
&z \geq 0
\end{aligned}$$



(a) Capacitated



(b) Uncapacitated

Figure 2.1: Uncapacitated version of Figure 1.1.

Again, we will use labels instead of dual variables (see Subsection 1.5.1). Let $\mu_i = 1/z_i$, where μ_i is called the label at node i. Then our dual program becomes:

$$\begin{aligned}
&\max \sum_{i \in V-t} \frac{b_i}{\mu_i} && \\
&\gamma_{ij}\frac{\mu_i}{\mu_j} \leq 1 && \forall ij \in E \\
&\mu_i > 0 && \forall i \in V - t \\
&\mu_t = 1 &&
\end{aligned}$$

We need to update the definition of excess for the supplies and demands. For the sink node t, we use the convention that $b_t = \infty$.

Definition 7. *The excess at a node $i \in V$ with respect to a flow f is $e_i(f) = \sum_{j:ji \in E} \gamma_{ji} f_{ji} - \sum_{j:ij \in E} f_{ij} - b_i$.*

When the context is unclear and we are comparing the excess between multiple flows, we will use the notation $e_i(f)$. When the context is clear, we will simply use e_i.

The complementary slackness conditions, with respect to the flow variables f and the labels μ, becomes

- Primal condition: $e_i = 0$ or $\frac{1}{\mu_i} = 0 \qquad \forall i \in V - t$
- Dual condition: $f_{ij} = 0$ or $\gamma_{ij} \frac{\mu_i}{\mu_j} = 1 \qquad \forall ij \in E$

However, since every node i has an arc to t, none of the μ_i can be ∞. Thus, the first condition must be satisfied by requiring $e_i = 0$ at all $i \neq t$.

In the second condition, $f_{ij} > 0$ implies that ij and its reverse arc ji are both in G_f. Since $\gamma_{ji} = 1/\gamma_{ij}$, the second condition is equivalent to saying that all arcs in the residual graph satisfy $\gamma_{ij} \frac{\mu_i}{\mu_j} \leq 1$. This is exactly the definition of conservative labels in Subsection 1.5.1, because the residual graph contains all of E and $\{ji : ij \in E, f_{ij} > 0\}$.

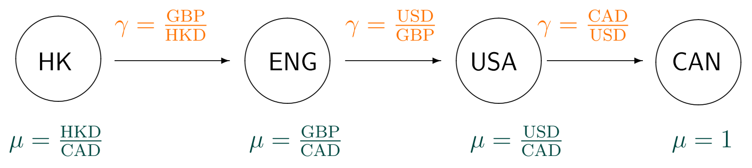
2.2 Labels

A set of labels μ can be used to relabel a network. The idea of relabelling a network is similar to working with reduced costs on the minimum cost flow problem.

2.2.1 Motivation Behind Labels: Converting Currencies

We discuss one motivation for relabeling a network by revisiting the foreign currency application (see Section 1.4). Consider a large corporation in Canada with a subsidiary in United States, which in turns owns a subsidiary in England, and the English company owns a subsidiary in Hong Kong. The three subsidiaries must send their profit (supply) to their respective parent company. Our goal is to determine how much money the Canadian corporation receives in the end. We can model this as a generalized flow, where the sink is the Canadian corporation. The gain factors would be the exchange rate between the countries of a subsidiary and its parent.

We cannot simply add up the profits together as the profits are in different currencies. We can find labels that represent the exchange rate between each country and Canadian dollars. Dividing the profits by the labels would adjust everything to Canadian dollars, and would allow us to add up all the profits earned by the Canadian corporation.



HK
ENG
USA
CAN
$\gamma = \frac{\text{GBP}}{\text{HKD}}$
$\gamma = \frac{\text{USD}}{\text{GBP}}$
$\gamma = \frac{\text{CAD}}{\text{USD}}$
$\mu = \frac{\text{HKD}}{\text{CAD}}$
$\mu = \frac{\text{GBP}}{\text{CAD}}$
$\mu = \frac{\text{USD}}{\text{CAD}}$
$\mu = 1$

Figure 2.2: Labels in currency exchange

2.2.2 A More General Look at Labels

How did we get the labels for our currency exchange example in Figure 2.2? If the residual graph contains no flow-generating cycles (see Section 1.5.2), then conservative labels can be found as follows [9]. Let P_i^* be a path of highest gain from i to t for each node $i \in V - t$.

$$\mu_i = \frac{1}{\prod_{kl \in P_i^*} \gamma_{kl}}$$

In the previous example, the labels μ represent the exchange rates from Canadian dollars. In general, if node i has a supply, then dividing the supply at node i by μ_i allows us to compute the maximum amount of flow that can be sent to sink t from node i. Given the equation $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$, if we divide both e_i and b_i by μ_i, then we must apply the same idea to the other terms. The equation expands as follows:

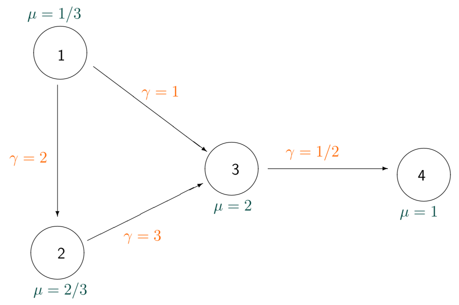
$$\frac{e_i}{\mu_i} = \sum_{ji \in E} \gamma_{ji} \frac{\mu_j}{\mu_i} \frac{f_{ji}}{\mu_j} - \sum_{ij \in E} \frac{f_{ij}}{\mu_i} - \frac{b_i}{\mu_i}$$

We can define our relabelled terms as follows:

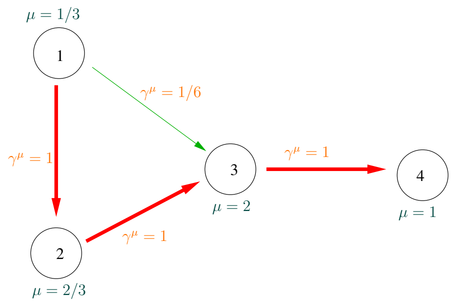
- Relabelled flow: $f_{ij}^{\mu} = f_{ij}/\mu_i$
- Relabelled demand/supply: $b_i^{\mu} = b_i/\mu_i$
- Relabelled excesses: $e_i^{\mu} = e_i/\mu_i$

- Relabelled gain factors: $\gamma_{ij}^{\mu} = \gamma_{ij}\frac{\mu_i}{\mu_j}$

Recall from Subsection 1.5.1 that an arc ij is tight if $\gamma_{ij}^{\mu} = 1$, using our relabelled notations. Furthermore, if P is a tight path from i to t, then P is the path of highest gain to send excesses from i. Even when we are sending flow between arbitrary nodes i and j, we should only send flow on a tight i-j path P'. P' is a segment of the highest gain path from i to t and thus would also be the highest gain path from i to j.



(a) Original flow: One unit leaving node 1 $\Rightarrow$ Three units entering node 4



(b) Relabelled flow: Three relabelled units leaving node 1 $\Rightarrow$ Three relabelled units entering node 4

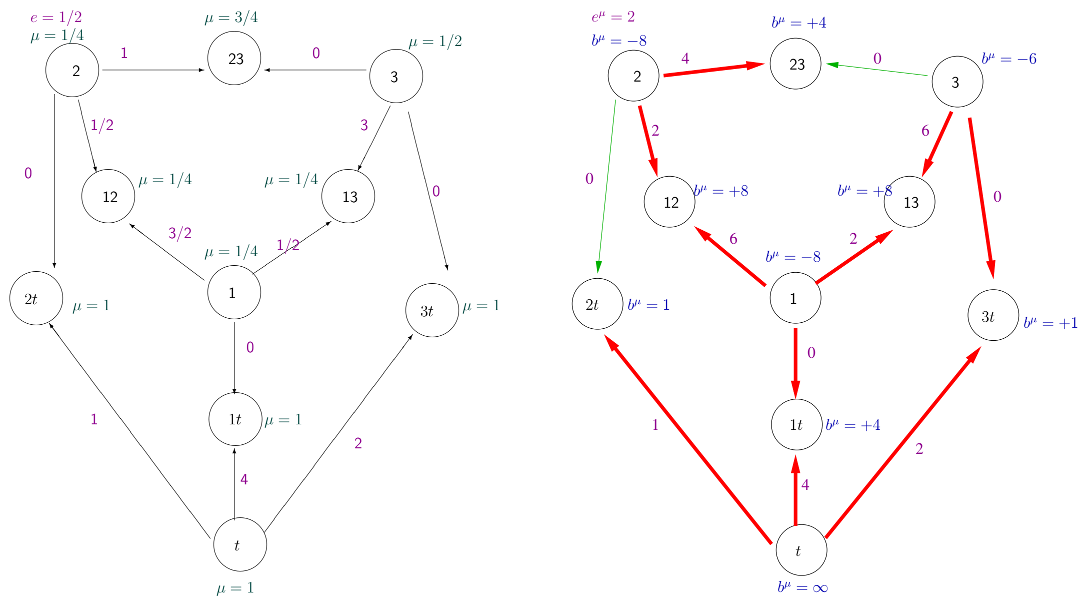
Figure 2.3: Simplicity in using labels; green arcs are non-tight and red arcs (bold) are tight.

In Figure 2.3a, it is not clear what amount of flow would arrive at node 4 if we try to send one real unit of flow from node 1. This would depend on which path we take to send the flow. Figure 2.3b shows the result of relabelling. Red arcs are tight and green arcs are non-tight. Sending one real unit of flow from node 1 is equivalent to sending three relabelled units. Furthermore, we know that the maximum flow that can arrive at node 4 is three relabelled units, the equivalent of three real units.

Moving back to the example from Figure 2.1, the original network with some initial flow and labels, as well as its relabelled network, are presented in Figure 2.5.

2.3 Δ-Fatness and Δ-Feasibility

As with previous scaling algorithms, Végh's algorithm maintains a pair of feasible primal and dual solutions for a given scaling parameter $\Delta \in \mathbb{R}$. The labels μ are conservative



(a) Feasible flow and labels. (b) Relabelled with μ.

Figure 2.4: Relabelling a network. f_{ij}, f^{μ}_{ij} are in purple (on the arcs). Non-tight arcs are green and tight arcs are red (bold) in (b).

Figure 2.5: Relabelling a network. f_{ij}, f^{μ}_{ij} are in purple (on the arcs). Non-tight arcs are green and tight arcs are red (bold) in (b).

only on arcs with a large amount of flow relative to Δ. In order words, we relax the dual optimality condition. The algorithm will update the primal and dual solutions for decreasing values of Δ until we arrive at an optimal pair.

In order to define which arcs have large flow relative to Δ, we need to define Δ-fat arcs. We can think of the Δ-fat graph as a subgraph of the residual graph, containing only those arcs where the residual capacity is more than Δ.

Definition 8. *The Δ-fat graph with respect to some Δ and f^{μ}, denoted $G^{\mu}_{f}(\Delta) = (V, E^{\mu}_{f}(\Delta))$, contains the arcs $E^{\mu}_{f}(\Delta) = E \cup \{ji : f^{\mu}_{ij} > \Delta\}$. An arc in this graph is said to be a Δ-fat arc.*

The second set of arcs in the above definition refers to the reverse arcs in a residual graph. For a reverse arc $ji \in E^{\mu}_{f}(\Delta)$, we will set $\gamma_{ji} = 1/\gamma_{ij}$.

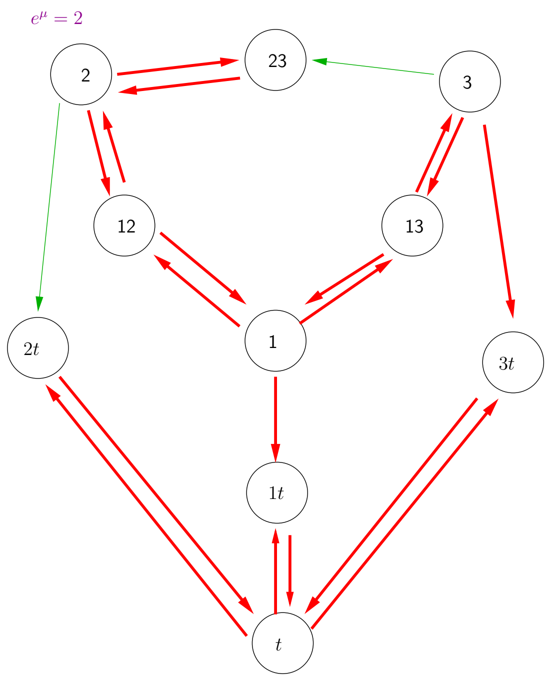


Figure 2.6: Δ-fat graph $G_f^\mu(\Delta)$ constructed from previous example in Figure 2.4b

Recall the complementary slackness condition from Section 2.1: $e_i = 0$ if $i \neq t$ and $\gamma_{ij}^\mu = 1$ if $f_{ij} > 0$. During the algorithm, we will only require arcs to be tight when they are Δ-fat. As Δ decreases, we will get closer to an optimal dual solution.

For the primal solution f, we will take a different approach. Instead of relaxing the optimality condition for the primal solution, we will require a stronger feasibility constraint and require each node to have a reserve of excess.

Definition 9. *The reserve at node i is $R_i = \sum_{ji:\gamma_{ji}^\mu<1} \gamma_{ji} f_{ji}$. This is the flow going into node i using non-tight arcs. Furthermore, we can also define the relabelled reserve as $R_i^\mu = R_i/\mu_i = \sum_{ji:\gamma_{ji}^\mu<1} \gamma_{ji}^\mu f_{ji}^\mu$.*

Notice that if our primal-dual pair (f, μ) is optimal, then all the non-tight arcs should have zero flow because of the second complementary slackness condition. Let i be any node in $V - t$. If $e_i \geq R_i$, then all flows into i on non-tight arcs could be "reset to zero" without violating primal feasibility. Removing all outflow from i on non-tight arcs can only increase the excess at i.

Definition 10. *A pair of solutions (f, μ) is a Δ-feasible pair if it satisfies the following three conditions:*

- *f and μ are both feasible*
- *$\gamma_{ij}^{\mu} \leq 1$ for all arcs in $E_f^{\mu}(\Delta)$*
- *$e_i \geq R_i$ for all $i \neq t$*

The first condition of Δ-feasibility is clear. Let us assume that it holds.

The second condition, which requires μ to be conservative only on the Δ-fat graph, implies that $\gamma_{ij}^{\mu} = 1$ whenever $f_{ij}^{\mu} > \Delta$ because both ij and its reverse arc ji are in $E_f^{\mu}(\Delta)$. This means that $\gamma_{ij}^{\mu} \leq 1$ and $\gamma_{ji}^{\mu} \leq 1$; therefore ij must be tight. For a node i, this also implies that $R_i^{\mu} \leq \deg(i)\Delta$ since all non-tight arcs have small relabelled flow, where $\deg(i)$ is the sum of in-degree and out-degree of node i.

The third condition allows us to reset the flow on non-tight arcs to zero without violating primal feasibility. Removing flow on non-tight arcs will mean that our pair (f, μ) satisfies dual optimality conditions completely, albeit not the primal optimality conditions.

2.4 High Excess and Low Excess Nodes: Sending Flow

When our scaling parameter is Δ, we will maintain the invariant that $e_i^{\mu} \leq 4(\deg(i) + 2)\Delta$ for all nodes $i \in V - t$.. The algorithm will run by sending excesses from nodes with high excess to nodes with low excesses, as we will see in Chapter 3. As a result, let us define the special sets of nodes that are crucial for both the weakly and strongly polynomial versions of Végh's algorithm.

Definition 11. *The set of high excess nodes is $T_0 \subseteq \{i : e_i^{\mu} \geq (\deg(i) + 2)\Delta\}$. Nodes are added to T_0 at one particular step of the algorithm, but they are removed immediately if the relabelled excess falls below $(\deg(i) + 2)\Delta$.*

Definition 12. *The set of low excess nodes is $L = \{i : e_i^{\mu} < (\deg(i) + 1)\Delta\}$.*

Since our sink t has $b_t = \infty$, its excess is always $e_t = -\infty$ and we must have $t \in L$. Furthermore, low excess nodes cannot become high excess nodes as a result of receiving flow.

As we described in Subsection 2.2.2, we always want to send flow along tight paths because this is the path of highest gain. This also keeps our (f, μ) Δ-feasible. If we send Δ units of relabelled flow from node $k \in T_0$ to $l \in L$ on a tight path P, then e_k^μ decreases by Δ and remains positive, e_l^μ increases, and all other relabelled excesses remain the same. So f is still a feasible flow. Our labels μ did not change and are feasible. For the second condition of Δ-feasibility, we could create a new arc ji to $E_f^\mu(\Delta)$ when we pushed flow on arc $ij \in P$. Since $\gamma_{ij}^\mu = 1$ on our tight path, we have $\gamma_{ji}^\mu = 1$. For the third condition, we do not change the reserve at any of the nodes since we do not use non-tight arcs, so we can simply consider the excess at node k. After augmenting flow, $e_k^\mu \geq (\deg(k) + 2)\Delta - \Delta > \deg(k)\Delta \geq R_k^\mu$. Thus all three conditions of Δ-feasibility are satisfied by our choice of thresholds in the definitions of T_0 and L.

We need to identify the set of nodes that are reachable from T_0 using tight paths:

Definition 13. *T denotes a subset of the set of nodes reachable from T_0. In detail, $T \subseteq \{i :$ There exists a tight path from some $j \in T_0$ to i in $E_f^\mu(\Delta)\}$.*

Our definition of T allows for trivial tight paths, so that all nodes in T_0 are also in T.

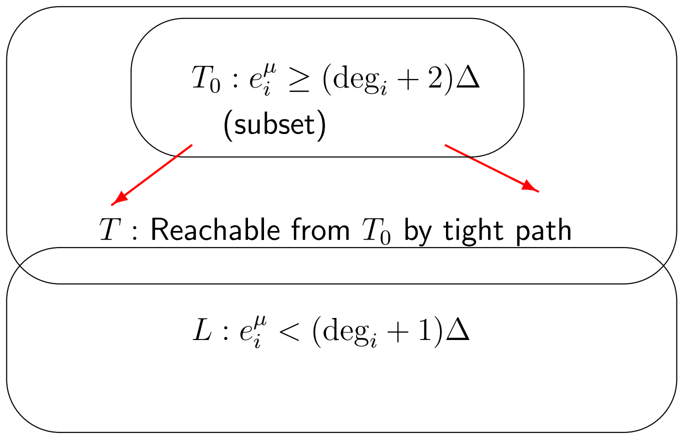

$T_0 : e_i^\mu \geq (\deg_i + 2)\Delta$
(subset)
T : Reachable from T_0 by tight path
$L : e_i^\mu < (\deg_i + 1)\Delta$

Figure 2.7: Relationship between sets T_0, T, and L

Thus if $T \cap L \neq \emptyset$, then there exists some low excess nodes reachable by the high excess nodes and we can send flow. Using the definition of T, we can divide E into 4 subsets:

- $E[T] = \{ij \in E : i, j \in T\}$
- $E[V \backslash T] = \{ij \in E : i, j \in V \backslash T\}$

- $E[T, V\backslash T] = \{ij \in E : i \in T, j \in V\backslash T\}$
- $E[V\backslash T, T] = \{ij \in E : i \in V\backslash T, j \in T\}$

2.5 Summary

In Chapter 2, we used an uncapacitated network to model the generalized flow problem by replacing capacities with supplies and demands at nodes. I used a foreign currency example to motivate the use of labels. Labels are used to relabel the network to the base unit of t, so that we can easily identify tight paths and determine the amount of flow being sent and received at nodes.

We discussed the optimality conditions for the uncapacitated model. An optimal primal-dual pair ensures that the excesses are 0 for all $i \in V - t$ and that all arcs with positive flow are tight. We also discussed the definitions of Δ-fat arcs and Δ-feasibility. The idea of Δ-feasibility is that we are relaxing the dual optimality conditions in order to send flow. On the other hand, we maintain a reserve of excess at nodes (that is, we have $e_i \geq R_i$, $\forall i \in V - t$) so that our algorithms can perform various updates without violating primal feasibility.

I also defined high excess, low excess, and reachable nodes. These nodes will be critical to running Végh's algorithm in Chapter 3. The idea is that we will move excesses from T_0 to L, but only on tight paths.

Notation	Meaning
Chapter 1: Introduction	
γ_{ij}	Gain factor of arc ij, always > 0
b_i	Demands (> 0) or supplies (< 0) at node i
e_i	Excess at node i, $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$
f_{ij}	Flow on arc ij
μ_i	Label at node i; inverse of dual solution, always > 0
E_f	The set of residual arcs: $\{ij : f_{ij} < u_{ij}\} \cup \{ji : f_{ij} > 0\}$
G_f	The residual graph with node set V and arc set E_f
Introduced in Chapter 2: Definitions and Notations	
f^μ_{ij}, b^μ_i, e^μ_i, γ^μ_{ij}	Relabelled quantities; see Subsection 2.2.2
$E^\mu_f(\Delta)$	The set of Δ-fat arcs: $E \cup \{ji : f^\mu_{ij} > \Delta\}$
$G^\mu_f(\Delta)$	The graph with node set V and arc set $E^\mu_f(\Delta)$
R_i	The reserve at i; $\sum_{ji:\gamma^\mu_{ji}<1} \gamma_{ji} f_{ji}$
T_0	High excess nodes; $T_0 \subseteq \{i : e^\mu_i \geq (\deg(i) + 2)\Delta\}$
T	Reachable nodes; $T \subseteq \{i : i \text{ is reachable from a node in } T_0 \text{ via a tight path in } E^\mu_f(\Delta)\}$
L	Low excess nodes; $\{i : e^\mu_i < (\deg(i) + 1)\Delta\}$

Table 2.1: Summary of Notations in Chapter 2

Chapter 3

Weakly Polynomial Algorithm

Since the strongly polynomial algorithm is quite complicated, I will first give a simpler version of the algorithm which runs in weakly polynomial time. The strongly polynomial version is an extension that incorporates the idea of abundant arcs and contractions.

A table of all the notations up to Chapter 3 is provided on page 52 at the end of the chapter. This will be useful for readers who are interested in a particular section rather than the full thesis.

3.1 Structure of the Algorithm

3.1.1 Main Algorithm

The algorithm works by maintaining a Δ-feasible pair (see Section 2.3). Unlike previous algorithms where we move positive excess directly to the sink, Végh's algorithm is slightly more relaxed in that we are allowed to move flow from high excess nodes in T_0 to low excess nodes in $T \cap L$ (see Section 2.4). The algorithm maintains the invariant that excesses are at most $4(\deg(i) + 2)\Delta$ for all $i \in V - t$. This invariant and the definition of Δ-feasibility means that we get closer and closer to an optimal pair of solutions as Δ gets smaller. Let us consider how we are able to achieve this goal. We start an iteration with a Δ-feasible pair (f, μ) and the sets T_0 and T.

Action 1: Augment Flow
In order to augment flow from a high excess node to a low excess node, we want to use

a tight path because it is the path of highest gain (see Subsection 2.2.2). We know that all the nodes in T are reachable from T_0 using tight paths in $E_f^{\mu}(\Delta)$, so we check for low excess nodes in T. If $T \cap L \neq \emptyset$, then we can send Δ units of relabelled flow from some node $i \in T_0$ to node $j \in T \cap L$.

Immediately after sending flow from node i to node j, it is possible that $e_i^{\mu} < (\deg(i)+2)\Delta$, which means that i no longer belongs in T_0. If so, the obvious correction is to remove node i from T_0. We should also reset T to T_0 because there could be nodes in T that are reachable from i only. As we will see later, Végh's algorithm will always reset T to T_0 after an augmentation in order to simplify the runtime analysis.

What if $T \cap L = \emptyset$? Let us consider other actions that we might perform when we cannot augment flow.

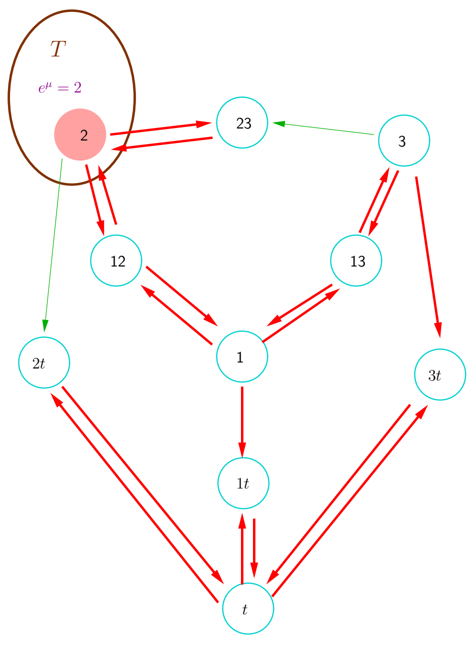
Action 2: Tight arc leaving T
Assume $T \neq \emptyset$. Since none of the nodes in T are low excess nodes, we want to check for more nodes that are reachable by T_0. To do this, we will look for tight arcs in $E_f^{\mu}(\Delta)$ leaving T. If such an arc ij exists, then node j is also reachable from T_0 and should be added into T. In the next iteration, we can look in the new T for low excess nodes.

Continuing our example from Figure 2.5, actions 1 and 2 are demonstrated on Figure 3.1 with $\Delta = 2/5$.

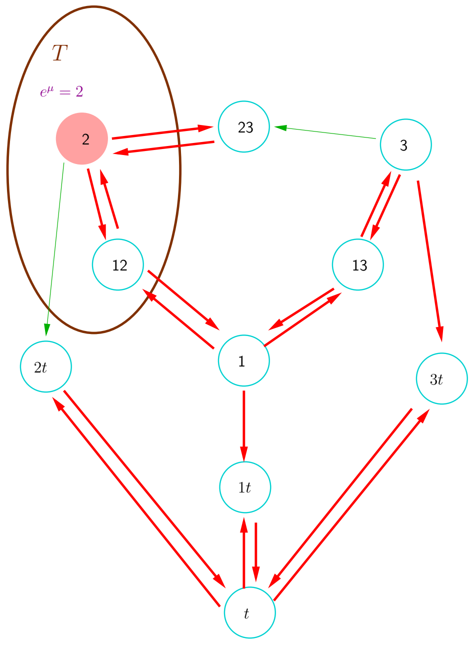
Action 3: None of the above
There are two reasons why we cannot perform the previous two actions. The first reason is that $T_0 = \emptyset$, which implies $T = \emptyset$. In this case, we need to raise the excesses on nodes relative to Δ without augmenting flow.

The second reason is that $T \neq \emptyset$, but $T \cap L = \emptyset$ and all arcs $ij \in E_f^{\mu}(\Delta)$ leaving T are non-tight. Notice that only original arcs are leaving T in $G_f^{\mu}(\Delta)$, otherwise arc ij and its reverse arc ji are both in $E_f^{\mu}(\Delta)$ and must be tight by the definition of Δ-feasibility. That is, $\{ij : ij \in E_f^{\mu}(\Delta), i \in T, j \in V \backslash T\} = E[T, V\backslash T]$. Thus we want to tighten an arc by increasing γ_{ij}^{μ} for some $ij \in E[T, V\backslash T]$.

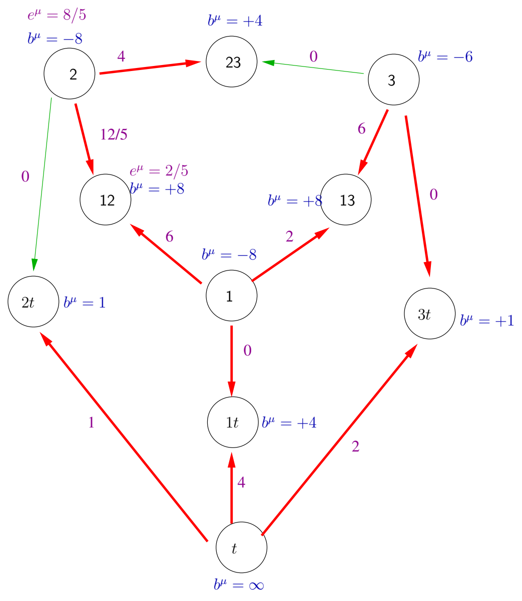
These obstructions to augmenting flows can be solved by modifying the labels μ or the scaling parameter Δ. We will call this procedure the Elementary Step.



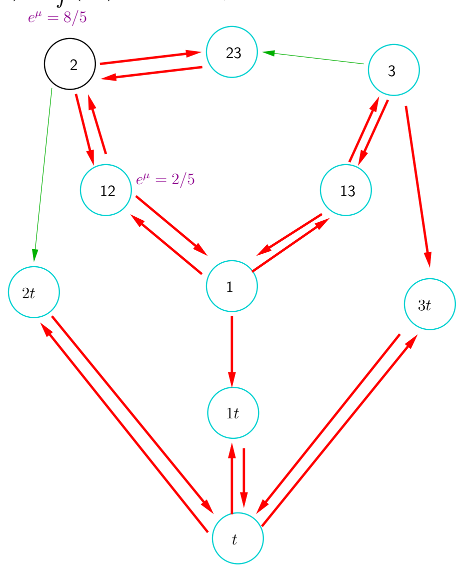
(a) $G_f^\mu(\Delta)$: $T_0 = \{2\}$. Expand T to proceed.



(b) $G_f^\mu(\Delta)$: $T \cap L \neq \emptyset \Rightarrow$ augment flow.



(c) G: New flow after augmenting along $P = 2, 12$. Compare with Figure 2.4a.



(d) $G_f^\mu(\Delta)$: After augmenting flow, $T = T_0 = \emptyset$. Node 2 is neither in T_0 nor L.

Figure 3.1: Depicting the progress of the algorithm with $\Delta = 2/5$. Pink nodes indicate $i \in T_0$ (shaded) and blue nodes indicate $i \in L$ in (a), (b), and (d). Red arcs (bold) are tight and green arcs are non-tight.

Algorithm 1: Main Algorithm, weakly polynomial version

```
1  Initialization: Find a $\Delta^{\text{start}}$-feasible pair $(f, \mu)$ using Tight-Flow, and set $T_0 = T = \emptyset$;
2  while $\Delta \geq \Delta^{\text{end}}$ do
3      Build $L = \{i : e_i^\mu < (\deg(i) + 1)\Delta\}$;
4      if $T \cap L \neq \emptyset$ then
5          Send $\Delta$ units of relabelled excess from $i \in T_0$ to $j \in L$ on a tight $i$-$j$ path in
           $E_f^\mu(\Delta)$;
6          if $e_i^\mu < (\deg(i) + 2)\Delta$ then remove $i$ from $T_0$;
7          Reset $T = T_0$;
8      else if $\exists$ tight arc $ij$ in $E[T, V \backslash T]$ then
9          $T = T \cup \{j\}$;
10     else
11         Proceed to Elementary Step
12     end
13 end
14 Termination: Perform Tight-Flow using optimal $\mu^*$;
```

I will leave the details of Initialization and the Tight-Flow subroutines until Section 3.4. The main idea is that Initialization will return solutions f and μ, as well as a starting scaling parameter Δ^{start} such that (f, μ) is Δ^{start}-feasible. It is also well known that if $\Delta < \Delta^{\text{end}}$ for some small Δ^{end} in a scaling algorithm, then we can find the optimal flow using one maximum flow computation [7].

Lines 3-7 corresponds to the first action of augmenting flow and doing the necessary updates to T_0 and T. Lines 8-9 corresponds to the second action of expanding T when there is a tight arc leaving T. If we cannot perform either action, we will move onto Elementary Step, which will be described next.

3.2 Elementary Step

Elementary Step should serve one of two goals: (1) Tighten an arc in $E[T, V \backslash T]$, or (2) add a node to T_0. Let us consider how each of these goals can be achieved.

3.2.1 Achieving Goals of Elementary Step

Goal 1: Tighten ij
First assume that $T \neq \emptyset$. Then for all arc $ij \in E[T, V\backslash T]$, the following holds with strict inequality:

$$\gamma_{ij}\frac{\mu_i}{\mu_j} < 1$$

The obvious action is to either increase label μ_i or to decrease label μ_j. We will choose the convention of only increasing labels. To tighten a specific ij, we can naïvely multiply μ_i by $\alpha_1 = 1/\gamma_{ij}^{\mu}$. This could adversely affect the other arcs in the network and cause some arcs to violate dual feasibility. We want to multiply all the labels in T by some $\alpha_1 > 1$ so that the relabelled gain factors are unchanged for all arcs that lie within T or within $V\backslash T$. Arcs in $E[V\backslash T, T]$ will be fixed later. Finally, because we are updating all the μ_i for $i \in T$, we need to ensure that our α_1 is not so large that some $\gamma_{ij}\frac{\alpha\mu_i}{\mu_j} > 1$ and violate dual feasibility. Thus, our update would be:

- Choose $\alpha_1 = \min\{1/\gamma_{ij}^{\mu} : ij \in E[T, V\backslash T]\}$.
- Set $\mu_i' = \alpha_1\mu_i$ for $i \in T$.
- Set $\mu_i' = \mu_i$ for $i \in V\backslash T$.

Goal 2: Add node i to T_0
We could also try adding nodes to T_0. This would occur if $T_0 = \emptyset$ or the update to add nodes is easier to achieve than tightening arcs. We will add a node i to T_0 when $e_i^{\mu} = 4(\deg(i)+2)\Delta$. This threshold is chosen partly to make the runtime analysis simpler. As our labels μ are strictly increasing, we cannot increase e_i^{μ} and must decrease Δ by some factor α_2 instead. The choice of α_2 is unclear, and will be discussed in Subsection 3.2.2. The desired update would be:

- Choose α_2 so that $e_i^{\mu} = 4(\deg(i)+2)\frac{\Delta}{\alpha}$ for some $i \in V\backslash T - t$ and $e_j^{\mu} \leq 4(\deg(j)+2)\frac{\Delta}{\alpha_2}$ for all $j \in V - t$.
- Set $\Delta' = \Delta/\alpha_2$.

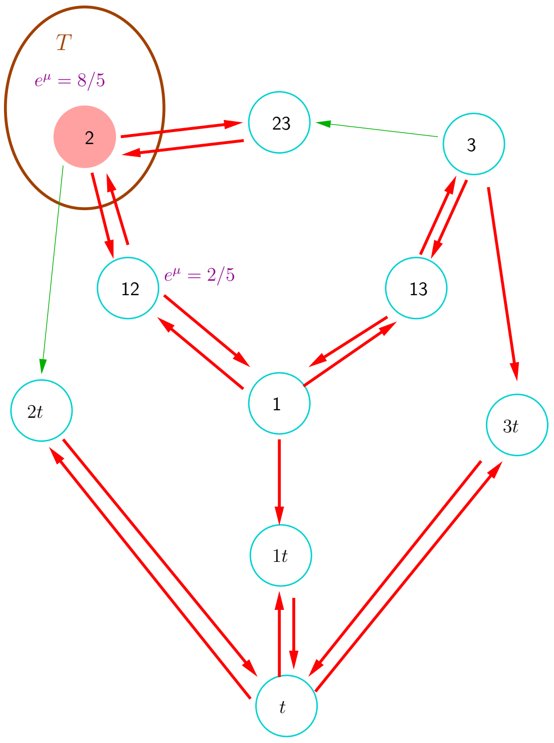


Figure 3.2: Proceeding from Figure 3.1, we get $\alpha = 5$ and $\Delta' = 2/25$ after Elementary Step. Initially, $T = \emptyset$ and no changes are made to the flow and to labels. After Elementary Step, $T_0 = T = \{2\}$.

Combining Goals 1 and 2

We want to combine the updates described into one subroutine, so that we update both the labels and the scaling parameter by the same α. Updating Δ clearly has no effect on whether arcs are tight or not. For node $i \in T$, updating both μ_i and Δ simultaneously means that we do not change the relationship between its relabelled excess and its scaling parameter. We only increase relabelled excess relative to the new scaling parameter for node $i \in V \backslash T$ because $\mu'_i = \mu_i$. The α that we will use for our update is the minimum of α_1 and α_2. In Figure 3.2, we see a demonstration of Elementary Procedure after we cannot augment flow or expand T in Figure 3.1.

The updated μ is feasible to the dual program. We know that sink t is in $V \backslash T$ when we proceed to Elementary Step, otherwise $t \in T \cap L$ and we could have augment flow. Therefore, we do not change t's label ($\mu'_t = 1$). Consider the first constraint in the dual program (see Section 2.1): $\gamma_{ij}^{\mu} \leq 1$. We do not change the left side of the inequality for ij in $E[T] \cup E[V \backslash T]$. For $ij \in E[T, V \backslash T]$, we multiplied the left side by $\alpha \leq 1/\gamma_{ij}^{\mu}$, so the inequality is still satisfied. For $ij \in E[V \backslash T, T]$, we increased the denominator in $\gamma_{ij}\frac{\mu_i}{\mu_j}$, and again the inequality is maintained. Thus μ is feasible.

We will skip the feasibility of the flow f because we still need to update f. One problem with our current set of updates is that we could potentially violate Δ-feasibility (see Definition 10). In particular, there could be an arc ij with $f_{ij}^{\mu} \leq \Delta$ and $\gamma_{ij}^{\mu} < 1$ before Elementary Step. Since its reverse arc ji is not Δ-fat, it was not in the $E_f^{\mu}(\Delta)$ and ij did not have to be tight. Our updates decrease Δ to Δ' and could result in $f'_{ij}/\mu'_i > \Delta'$, causing both ij and ji to become Δ'-fat. We need to either tighten ij or modify the flow on ij so that it is not Δ'-fat in order for (f', μ') to be a Δ'-feasible pair.

Regaining Δ'-feasibility
Consider an arc ij with $i \in T$. Tight arcs with node $i \in T$ stay tight because node j is also in T. For the non-tight arcs, we start with $f_{ij}/\mu_i \leq \Delta$ and our label updates ensure that $f'_{ij}/\mu'_i = f_{ij}/(\alpha\mu_i) \leq \Delta/\alpha = \Delta'$. So these arcs do not violate Δ'-feasibility.

Now let us consider an arc with node $i \in V \backslash T$. Tight arcs inside $V \backslash T$ stay tight, so the problem lies with arcs in $E[V \backslash T, T]$ and non-tight arcs in $E[V \backslash T]$. First, notice that for arc $ij \in E[V \backslash T, T]$, we must have $f_{ij}^{\mu} \leq \Delta$. Otherwise the reverse arc ji is also in $E_f^{\mu}(\Delta)$ and both arcs must be tight. This means that we should add node i to T rather than proceed to Elementary Step. The non-tight arcs in $E[V \backslash T]$ also satisfies $f_{ij}^{\mu} \leq \Delta$. As $\mu'_i = \mu_i$, the only other update that we can do is to set $f'_{ij} = f_{ij}/\alpha$ on these arcs. Then we achieve $f'_{ij}/\mu'_i = (f_{ij}/\alpha)/\mu_i \leq \Delta/\alpha$.

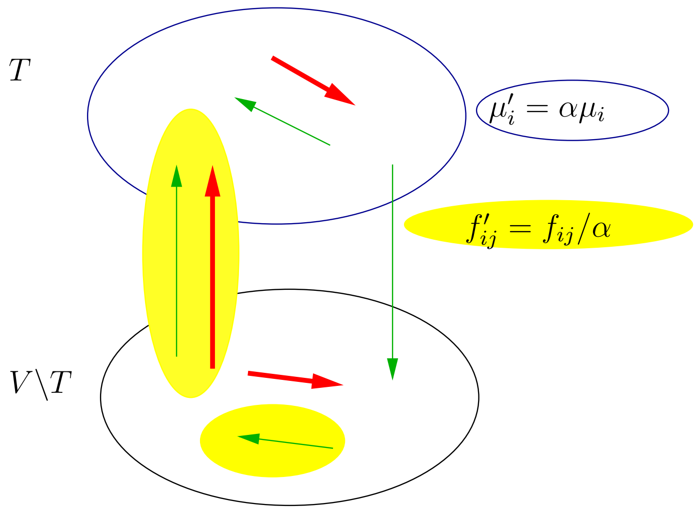


Figure 3.3: Flow and labels are updated in Elementary Step; red arcs (bold) are tight and green arcs are non-tight. We update labels for nodes in T and flows on the highlighted arcs.

Finally, we need to check that $e'_i \geq R'_i$. In Lemma 15 of Subsection 3.2.2, we will see that this condition is achieved with our current updates.

These updates to the labels and flows are summarized in Figure 3.3.

Concluding Elementary Step

After performing all the updates, we achieve one of the two goals. If there is a node $i \in V\backslash T$ such that $e_i'^{\mu'} = 4(\deg(i) + 2)\Delta'$, then we need to add i to T_0. Because every node in T_0 must also be in T, we can update T by taking $T = T \cup T_0$. Note that we do not need to update T for new nodes reachable by new tight arcs because this update can be performed in the next iteration. Finally, because we reduced the inflow into nodes in T with our flow update, it is possible that the relabelled excess could have dropped below $(\deg(i) + 2)\Delta'$ for some node $i \in T_0$, and we would need to remove node i. Similar to the update after augmentation, we need to reset $T = T_0$ whenever we remove a node from T_0. The difference is that we reset T only if T_0 loses a node, rather than always reset T.

Updating both Δ and μ by the same factor α is a new idea and called continuous scaling by Végh. Continuous scaling makes it easier for us to correct our solution pair so that it remains Δ-feasible after the Elementary Step. The Elementary Step proceeds as follows:

```
Algorithm 2: Elementary Step
1  Compute $\alpha_1$ = minimum value to tighten an arc in $E[T, V\backslash T]$;
2  Compute $\alpha_2$ = minimum value to raise a node's relabelled excess to $4(\deg(i)+2)\Delta'$;
3  Set $\alpha = \min\{\alpha_1, \alpha_2\}$;
4  Set $\Delta' = \Delta/\alpha$;
5  for $i \in V$ do
6      if $i \in T$ then $\mu'_i = \alpha\mu_i$;
7      if $i \in V\backslash T$ then $\mu'_i = \mu_i$;
8  end
9  for $ij \in E$ do
10     if $ij \in E[V\backslash T, T]$ then $f'_{ij} = f_{ij}/\alpha$;
11     else if $ij \in E[V\backslash T]$ and $\gamma^{\mu'}_{ij} < 1$ then $f'_{ij} = f_{ij}/\alpha$;
12     else $f'_{ij} = f_{ij}$;
13 end
14 Build up $T_0 = T_0 \cup \{i : i \in V\backslash T, e'^{\mu'}_i = 4(\deg(i)+2)\Delta'\}$;
15 Update $T = T \cup T_0$ ;
16 if $\exists i \in T_0 : e'^{\mu'}_i < (\deg(i)+2)\Delta'$ then
17     Remove $i$ from $T_0$;
18     Reset $T = T_0$;
19 end
```

The first for-loop (lines 5-8) will update the labels. Th second for-loop (lines 9-13) will update the flows. Finally, lines 14-19 performs the concluding updates to ensure that the definition of T_0 is satisfied.

3.2.2 Finding α and Maintaining Δ'-feasibility

When we enter Elementary Step, the following conditions hold for our Δ-feasible pair (f, μ):

(i) $e^{\mu}_i < 4(\deg(i)+2)\Delta$ for all $i \in V\backslash T$.

(ii) $e^{\mu}_i \geq (\deg(i)+1)\Delta$ for all $i \in T$.

(iii) $\gamma^{\mu}_{ij} < 1$ for all $ij \in E[T, V\backslash T]$.

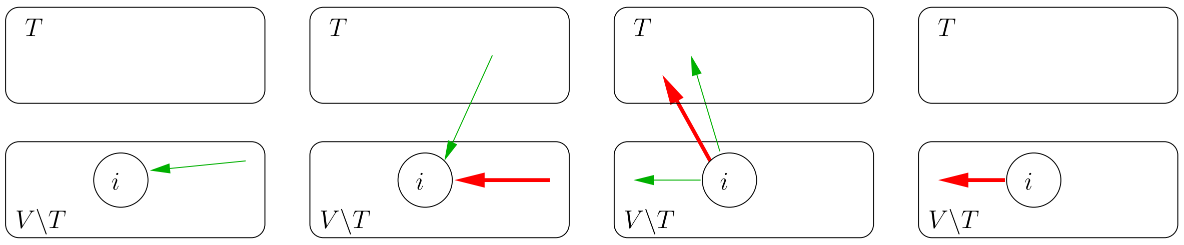
(iv) $f_{ij}^{\mu} \leq \Delta$ for all $ij \in E[V \backslash T, T]$.

Let us see why (i) must hold. Assume by contradiction that there exists $i \in V \backslash T$ such that $e_i^{\mu} = 4(\deg(i) + 2)\Delta$. Relabelled excesses cannot increase past $(\deg(i) + 2)\Delta$ outside Elementary Step by the definition of low excess nodes (see Section 2.4). The choice of Δ^{start} (see Section 3.4) means that we cannot have this with equality at $\Delta = \Delta^{\text{start}}$ and thus we should have gone through at least one Elementary Step. Then i should have been added to T_0 at the previous Elementary Step and should have remained there until its excess drops.

We know that (ii) must hold, otherwise we can augment flow. Next, (iii) must hold because we cannot expand T. As discussed previously, if (iv) does not hold, then $ji \in E_f^{\mu}(\Delta)$ and ij must be tight. This implies that i should be added to T instead of proceeding to Elementary Step.

To tighten an arc in $E[T, V \backslash T]$ without violating dual feasibility, choose $\alpha_1 = \min\{1/\gamma_{ij}^{\mu} : ij \in E[T, V \backslash T]\}$ (see Subsection 3.2.1). Clearly, $1 < \alpha_1 < \infty$ because there exists at least one arc to $t \in V \backslash T$. The computation for α_1 is quite simple. Given that $\gamma_{ij}^{\mu} < 1$ for all $ij \in E[T, V \backslash T]$, we simply need to multiply these terms by the smallest value amongst $1/\gamma_{ij}^{\mu}$ for ij going across the cut to ensure that we do not increase the relabelled gains above 1. Thus, $\alpha_1 = \min\{1/\gamma_{ij}^{\mu} : ij \in E[T, V \backslash T]\} > 1$.

The computation for α_2 is more complicated. To ensure that $e_i'^{\mu'} = 4(\deg(i) + 2)\Delta'$ for some $i \in V \backslash T$ after all the updates in Elementary Step, we should determine α_2 by considering the updated relabelled excess. First, let us partition the arcs incident to $i \in V \backslash T$ into four types:



(a) $F_1(i)$ (b) $F_2(i)$ (c) $F_3(i)$ (d) $F_4(i)$

Figure 3.4: Partitioning arcs to compute α_2; red arcs (bold) are tight and green arcs are non-tight.

Set	Flow related with the set
$F_1(i) = \{ji : \gamma^{\mu}_{ji} < 1, ji \in E[V\backslash T]\}$	$r_1(i)_= \sum_{ji \in F_1(i)} \gamma_{ji} f_{ji}$
$F_2(i) = \{ji : \gamma^{\mu}_{ji} = 1, ji \in E[V\backslash T]\} \cup \{ji : ji \in E[T, V\backslash T]\}$	$r_2(i)_= \sum_{ji \in F_2(i)} \gamma_{ji} f_{ji}$
$F_3(i) = \{ij : \gamma^{\mu}_{ij} < 1, ij \in E[V\backslash T]\} \cup \{ij : ij \in E[V\backslash T, T]\}$	$r_3(i)_= \sum_{ij \in F_3(i)} f_{ij}$
$F_4(i) = \{ij : \gamma^{\mu}_{ij} = 1, ij \in E[V\backslash T]\}$	$r_4(i)_= \sum_{ij \in F_4(i)} f_{ij}$

Table 3.1: Partitioning arcs to compute α_2

The partition tells us that both F_1 and F_3 are updated in the Elementary Step. The new relabelled excess should satisfy:

$$\frac{e'_i}{\mu'_i} = \frac{\frac{r_1(i)}{\alpha_2} + r_2(i) - \frac{r_3(i)}{\alpha_2} - r_4(i)}{\mu_i} \leq 4(\deg(i) + 2)\frac{\Delta}{\alpha_2}$$

$$\alpha_2 = \min\{\frac{4(\deg(i) + 2)\Delta\mu_i + r_3(i) - r_1(i)}{r_2(i) - r_4(i) - b_i} : i \in V\backslash T\}$$

If the denominator is 0 for all nodes $i \in V\backslash T$, we will say that $\alpha_2 = \infty$. We also know that $\alpha_2 > 1$ by rearranging the following inequality:

$$\frac{e_i}{\mu_i} = \frac{r_1(i) + r_2(i) - r_3(i) - r_4(i) - b_i}{\mu_i} \leq 4(\deg(i) + 2)\Delta$$

Claim 14. *Elementary Step will compute an α such that $1 < \alpha < \infty$ and at least one of the following will happen:*

- *An arc ij in $E[T, V\backslash T]$ is now tight.*
- *A node $i \in V\backslash T$ satisfies $e'^{\mu'}_i = 4(\deg(i) + 2)\Delta'$.*

Proof. This follows from the minimality of α_1, α_2 and the way that both terms were computed. □

Lemma 15. *Elementary Step will return a Δ'-feasible pair (f', μ')*

Proof. To show Δ'-feasibility, I will show that condition 2 and condition 3 of Definition 10 holds. That is, all arcs in $E_{f'}^{\mu'}(\Delta')$ satisfy $\gamma_{ij}^{\mu'} \leq 1$ and $e'_i \geq R'_i$ for all $i \neq t$. These two conditions implies the new flow and labels are feasible (condition 1). Recall that the reserve R_i is the incoming flow on non-tight arcs to i (see Definition 9).

Condition 2: Arcs in $E_{f'}^{\mu'}(\Delta')$ satisfy $\gamma_{ij}^{\mu'} \leq 1$
This is due to the flow updates that we chose to perform (see Figure 3.3 and the preceding argument).

Condition 3: $e_i \geq R_i$ for all $i \neq t$
For nodes in $V\backslash T$, the only decrease on incoming flow is on non-tight arcs. Thus the decrease on e_i and R_i are the same during the flow modification step. We could also change the outflow of i, but this only increases e_i without affecting R_i.

For nodes in T, Elementary Step only modifies the flow coming in from $V\backslash T$. Since $\alpha > 1$, all the arcs in $E[V\backslash T, T]$ are now non-tight. Consider the change in excess:

$$\begin{aligned} e'_i &= e_i - \sum_{ji \in E[V\backslash T,T]} (\gamma_{ji} f_{ji} - \gamma_{ji} f'_{ji}) \\ &\geq e_i - \sum_{ji \in E[V\backslash T,T]} (\Delta \mu_i - \gamma_{ji} f'_{ji}) \end{aligned}$$

The second line is true because $f_{ji}/\mu_j \leq \Delta$ for all $ji \in E[V\backslash T, T]$ and $\gamma_{ji}\frac{\mu_j}{\mu_i} \leq 1$. This gives us $\gamma_{ji} f_{ji} \leq \Delta \mu_i$. Let λ_i denote the number of arcs $ji \in E[V\backslash T, T]$. We also know that $e_i/\mu_i \geq (\deg(i) + 1)\Delta$ since we did not augment flow to i.

$$\begin{aligned} e'_i &\geq e_i - \sum_{ji \in E[V\backslash T,T]} (\Delta \mu_i - \gamma_{ji} f'_{ji}) \\ &\geq (\deg(i) + 1)\Delta \mu_i - \lambda_i \Delta \mu_i + \sum_{ji \in E[V\backslash T,T]} \gamma_{ji} f'_{ji} \\ &= (\deg(i) + 1 - \lambda_i)\Delta \mu_i + \sum_{ji \in E[V\backslash T,T]} \gamma_{ji} f'_{ji} > R'_i \end{aligned}$$

We know that there are at most $(\deg(i) - \lambda_i)$ non-tight arcs $ji \in E[T]$, each contributing inflow of at most $\gamma_{ji} f'_{ji} \leq \Delta' \mu'_i$ by the same argument as above. The remaining non-tight arcs are in $E[V\backslash T, T]$. Thus the last inequality is true. □

3.3 Runtime Analysis

Since our focus is the strongly polynomial algorithm, in this section I will only present some of the proofs needed for the runtime analysis. Rather, the focus is on the structure of the proof as the same setup will be used when we discuss the runtime analysis for the strongly polynomial algorithm.

The runtime analysis for the weakly polynomial algorithm has two main components, which will be defined in the following subsections:

- A potential function Φ; and
- A partition of the iterations into three types: shrinking, expanding, and neutral.

Φ will be a non-negative function that decreases by 1 whenever we perform a shrinking iteration. If we can bound the total increase in Φ by some value Q, then we know that there are at most Q shrinking iterations. That is, if $\Phi^{(\theta)}$ is the value of Φ in iteration θ, then Q is:

$$Q = \sum_{\text{iterations } \theta} \max\{\Phi^{(\theta+1)} - \Phi^{(\theta)}, 0\}$$

An example of a shrinking iteration is a flow augmentation. This suggests that Φ should be related to e_i^μ/Δ because e_i^μ decreases by Δ when i sends flow. The iterations will be defined so that the number of expanding and neutral iterations can be bounded as a polynomial of the number of shrinking iterations (see Subsection 3.3.1).

Our potential function is:

$$\Phi = \sum_{i \in T_0} \left(\left\lfloor \frac{e_i^\mu}{\Delta} \right\rfloor - (\deg(i) + 1) \right)$$

The value of Φ starts at 0, because T_0 starts at $\emptyset$. Φ is always non-negative, because $e_i^\mu \geq (\deg(i) + 2)\Delta$ always holds for $i \in T_0$. Thus each node in T_0 contributes at least 1 to Φ. I will show that we can bound the total number of iterations based on Q. This analysis is common to both the weakly and strongly polynomial algorithm, and will be presented in full details. I will then state, without proof, the maximum value of Q for the weakly polynomial algorithm.

3.3.1 Bounding Shrinking, Expanding, and Neutral Iterations

Let $T^{(\theta)}$ represent the set T at the beginning of iteration θ. The iterations of the algorithm can be divided into three types based on the change in T:

- Shrinking - $T^{(\theta)} \backslash T^{(\theta+1)} \neq \emptyset$. That is, T loses at least one node, but can gain new nodes.
- Expanding - $T^{(\theta)} \subset T^{(\theta+1)}$.
- Neutral - Not shrinking nor expanding.

Lemma 16. *Given Q, the total number of iterations in the algorithm is $2nQ$.*

Proof. First, I will show that we have at most Q shrinking iterations. A shrinking iteration can occur in two ways: either when we augment flow from T_0 or when we remove nodes from T_0 in the Elementary Step. After we augment flow from node $i \in T_0$ to node $j \in T \cap L$, we reset $T = T_0$ and node j is removed from T. At the same time, either e_i^{μ} decreases by Δ if node i stays in T_0 or i is removed from T_0. Both cases decrease Φ by 1 because each term in Φ is at least 1 at the beginning of the iteration. Similarly, if the relabelled excess of a node $i \in T_0$ falls below the $(\deg(i) + 2)\Delta$ threshold and is removed during Elementary Step, then i is removed from T_0 and T shrinks. This also decreases Φ by 1. Since every shrinking iteration decreases Φ by 1, we have at most Q shrinking iterations.

There are at most n expanding iterations between two shrinking iterations because we are bounded by the number of nodes.

Finally, consider a neutral iteration θ. Neutral iterations can only happen in the Elementary Step, as the main algorithm shrinks T when we augment flow or expands T when we find tight arcs leaving T. In Elementary Step, we cannot remove nodes from T_0 because we would also remove nodes from T (steps 16-18) and result in a shrinking iteration. Now consider the two choices of α. If $\alpha = \alpha_2$ so that we add a node to T_0, then T grows by one node and this becomes an expanding iteration. If $\alpha = \alpha_1$, then we tighten an arc in $E[T, V \backslash T]$ and do not change T. Thus the only action that Elementary Step can perform is to tighten an arc going from T to $V \backslash T$. In iteration $\theta + 1$, we can either augment flow or expand T. This implies that every neutral iteration is followed by a shrinking iteration or an expanding iteration, and thus there are at most n neutral iterations between two shrinking iterations.

Since there are at most Q shrinking iterations and at most $2n$ iterations between every two shrinking iterations, there are at most $2nQ$ iterations. $\square$

3.3.2 Total Number of Iterations

The challenge is to find an upper bound on Q. Since excesses of nodes in T_0 can only decrease, we can only increase Φ when we add a node to T_0.

Theorem 17. *During the algorithm, each node i enters T_0 at most $\log \frac{\Delta^{\text{start}}}{\Delta^{\text{end}}}$ times. As a result, $Q \leq 13m \log \frac{\Delta^{\text{start}}}{\Delta^{\text{end}}}$.*

The proof of the first statement is omitted as we will see this bound computed for the strongly polynomial algorithm. I will show the proof of the second statement as the proof of the strongly polynomial algorithm follows the same structure.

Proof of second statement in Theorem 17. When a node i enters T_0 through the Elementary Step, its relabelled excess is exactly $4(\deg(i) + 2)\Delta$. Thus it contributes $4(\deg(i) + 2) - (\deg(i) + 1) = 3\deg(i) + 7$ to Φ. Given that a node i enters T_0 a limited number of times, we can bound:

$$Q \leq \sum_{i \in V-t} \log \frac{\Delta^{\text{start}}}{\Delta^{\text{end}}} (3\deg(i) + 7) \leq (6m + 7n) \log \frac{\Delta^{\text{start}}}{\Delta^{\text{end}}} \leq 13m \log \frac{\Delta^{\text{start}}}{\Delta^{\text{end}}}$$

□

Combining Lemma 16 and Theorem 17 implies that the total number of iterations is at most $26mn \log \frac{\Delta^{\text{start}}}{\Delta^{\text{end}}}$. Both Δ^{start} and Δ^{end} are chosen to be polynomials of n, m, and the bit sizes of numerical input. Thus, the total number of iterations is weakly polynomial.

3.4 Initialization and Termination

3.4.1 Tight-Flow on V, μ

The Tight-Flow subroutine is used both at Initialization and Termination. In order to run Tight-Flow, we need some labels μ. The idea is to send as much excess to t as possible, using the tight arcs identified thus far. Since we are only working on tight arcs, we can use relabelled quantities and treat the problem as a traditional flow problem. If we give t a demand of 0, then we are simply computing a maximum flow into t subject to non-negative excesses.

Algorithm 3: Tight-Flow at Initialization and Termination

```
1 Tight-Flow on $(V, \mu)$: begin
2     Construct subgraph with nodes $V$.
3     Add all tight arcs with both ends in $V$.
4     All arcs get capacities $u_{ij} = \infty$.
5     All nodes except $t$ gets supplies/demands $b_i^\mu$. The sink $t$ gets demand of 0.
6     Compute $g$: A maximum flow into $t$ such that all nodes have non-negative
      excesses.
7 end
```

Define our flow f so that $f_{ij} = g_{ij}\mu_i$ for the original network. We can define the flow f using the above relationship because g is simply the relabelled flow on a relabelled network.

3.4.2 Initialization

To initialize a starting pair (f, μ), we first compute an initial feasible flow. To do this, simply map the zero-flow on the capacitated network to the uncapacitated network. Next, we want to remove all flow-generating cycles on the residual graph using Radzik's maximum-mean-gain-cycle-cancelling algorithm (see Subsection 1.5.2, Section 1.6). By ensuring that there are no flow-generating cycles, we will be able to find conservative labels μ, albeit creating excesses at nodes.

We want to redefine a flow f before we start our algorithm. To do this, run Tight-Flow with our μ so that all positive flows are on tight arcs. Notice that we satisfy the dual optimality condition but not the primal optimality condition. We will also set $\Delta^{\text{start}} = \max_{i \in V-t} e_i^\mu$ as the first scaling parameter. This ensures that $e_i^\mu \leq ((\deg(i) + 2)\Delta^{\text{start}})$ for all $i \in V$ so that we can initialize $T_0 = \emptyset$. Our choice of Δ^{start} also ensures that (f, μ) is a Δ^{start}-feasible pair (see Definition 10). Clearly f and μ are feasible to the primal and dual respectively. Condition 2 is satisfied because we only have positive flow on tight arcs. Condition 3 is satisfied because $R_i = 0$ and our feasible flow f implies that $e_i \geq 0$.

3.4.3 Termination

Let $\Delta^{\text{end}} = 1/(17mU^3)$ for some large U defined in Chapter 6. When the algorithm terminates at $\Delta < \Delta^{\text{end}}$, we have a Δ-feasible pair (f, μ) (which is also Δ^{end}-feasible), but f might not be optimal. An optimal flow can be found by running Tight-Flow(V, μ).

Lemma 18. *Let (f, μ) be a Δ-feasible pair for some $\Delta < \Delta^{\text{end}}$. Then Tight-Flow$(V, \mu)$ returns an optimal primal-dual pair (f', μ).*

Proof. Our algorithm ensures that $e_i^\mu \leq 4(\deg(i) + 2)\Delta$ for all $i \in V - t$.

$$\sum_{i \in V-t} e_i^\mu(f) \leq \sum_{i \in V-t} 4(\deg(i) + 2)\Delta < 8(m+n)\Delta^{\text{end}} \leq 16m\Delta^{\text{end}}$$

Now define a new flow $\tilde{f}$ such that $\tilde{f}_{ij} = 0$ if arc ij is not tight and $\tilde{f}_{ij} = f_{ij}$ if arc ij is tight. Since $\tilde{f}$ is a feasible flow using only the tight arcs, we know that $\tilde{f}^\mu$ is a feasible solution to Tight-Flow (V, μ). Also, $e_i \geq R_i = 0$ implies that $(\tilde{f}, \mu)$ is Δ- feasible. Since $f_{ij}^\mu \leq \Delta$ when arc ij is non-tight, this means that we removed at most $m\Delta$ units of relabelled excess when we constructed $\tilde{f}^\mu$. Given that $\Delta < \Delta^{\text{end}} = 1/(17mU^3)$, we get the following inequality:

$$\sum_{i \in V-t} e_i^\mu(\tilde{f}) \leq \sum_{i \in V-t} e_i^\mu(f) + m\Delta < 17m\Delta^{\text{end}} = \frac{1}{U^3}$$

Let g be the resulting flow when we run Tight-Flow(V, μ). We can define a new flow f' on the generalized network by $f_{ij}'^\mu = g_{ij}$. I will show that $\sum_{i \in V-t} e_i^\mu(f') < 1/(17mU^3)$, which will help us show that f' is an optimal flow.

The following equation is true because summing the net flow (i.e. inflow less outflow) over all nodes is always zero.

$$\sum_{i \in V}\left(\sum_{ji \in E} f_{ji}'^\mu - \sum_{ij \in E} f_{ij}'^\mu\right) - \sum_{i \in V-t} b_i^\mu = \sum_{i \in V}\left(\sum_{ji \in E} \tilde{f}_{ji}^\mu - \sum_{ij \in E} \tilde{f}_{ij}^\mu\right) - \sum_{i \in V-t} b_i^\mu$$

We can rewrite the previous equation using the excess notation.

$$\sum_{i \in V-t} e_i^\mu(f') + \sum_{jt \in E} f_{jt}'^\mu - \sum_{tj \in E} f_{tj}'^\mu = \sum_{i \in V-t} e_i^\mu(\tilde{f}) + \sum_{jt \in E} \tilde{f}_{jt}^\mu - \sum_{tj \in E} \tilde{f}_{tj}^\mu$$

Since $g = f'^\mu$ is a maximum flow and $\tilde{f}^\mu$ is a feasible flow on Tight-Flow(V, μ), we know that $\sum_{jt \in E} f_{jt}'^\mu - \sum_{tj \in E} f_{tj}'^\mu \geq \sum_{jt \in E} \tilde{f}_{jt}^\mu - \sum_{tj \in E} \tilde{f}_{tj}^\mu$. This implies that $\sum_{i \in V-t} e_i^\mu(f') \leq \sum_{i \in V-t} e_i^\mu(\tilde{f}) < 1/U^3$.

To show that f' is an optimal flow, let us look at the complementary slackness conditions with f' and μ (see Section 2.1). We know that arcs ij with $f_{ij}' > 0$ are all tight arcs. So we need to show that $e_i = 0$ for all $i \in V - t$. By contradiction, assume there exists $k \in V - t$ such $e_k > 0$. Let Z be the set of nodes that are reachable from k using arcs in the residual graph of f'. Consider the excesses of nodes in Z (see explanations below).

$$\begin{aligned}
e_k^\mu(f') &\leq \sum_{i \in Z} e_i^\mu(f') \\
&= \sum_{i \in Z} \left(\sum_{ji \in E} f'^\mu_{ji} - \sum_{ij \in E} f'^\mu_{ij} - b_i^\mu \right) \\
&= \sum_{i \in Z} \left(\sum_{\substack{ji \in E: \\ j \in Z}} f'^\mu_{ji} - \sum_{\substack{ij \in E: \\ j \in Z}} f'^\mu_{ij} \right) - \sum_{i \in Z} b_i^\mu \\
&= -\sum_{i \in Z} b_i^\mu \\
&= -\frac{1}{\mu_k} \sum_{i \in Z} b_i \frac{\mu_k}{\mu_i}
\end{aligned}$$

Clearly sink $t \notin Z$, otherwise we can increase the flow to t by pushing some ϵ amount from k. Furthermore, there is no original arc leaving Z. In the second and third lines above, this means that if we have an arc ij with node $i \in Z$, then node $j \in Z$. Moreover, any original arc ji entering Z has zero flow, otherwise we have a residual arc ij leaving Z. This means that $f'^\mu_{ji} = 0$ if $j \notin Z$ and gives us the third line.

Chapter 6 defines U to be 2 times the product of all the numerators and denominators used to represent supplies/demands b_i and gain factors γ_{ij}. So b_i is an integer multiple of $1/U$. Since there is a tight path P_i from node k to node $i \in Z$, we know that $\mu_k/\mu_i = \prod_{e \in P_i} 1/\gamma_e$. Thus this term is also an integer multiple of $1/U$ and $\sum_{i \in Z} b_i \mu_k/\mu_i$ is an integer multiple of $1/U^2$.

Lastly, we assumed that there is an arc $kt \in E$, where $\gamma_{kt} \geq 1/U$. By feasibility of μ, we know that $\gamma^\mu_{kt} \leq 1$. This implies that $1/U \leq 1/\mu_k$, and the last term of the above inequality must be at least $1/U^3$ whenever it is positive.

$$\sum_{i \in V - t} e_i^\mu(f') \geq \sum_{i \in Z} e_i^\mu(f') \geq 1/U^3$$

This is a contradiction to $\sum_{i \in V - t} e_i^\mu(f') < 1/U^3$. $\square$

3.5 Summary

In this chapter, I introduced the weakly polynomial version of Végh's algorithm. In the algorithm, we want to augment flow from high excess T_0 to low excess nodes L. Failing this, we will try to expand the set of reachable nodes T in hopes of reaching a low excess node by tight arcs. If we cannot do either, then we will update the scaling parameter Δ and the labels μ. In contrast to previous algorithms, both Δ and μ are scaled by the same factor α. The choice of α will allow us to either expand the set of high excess nodes T_0 or tighten an arc in $E[T, V \backslash T]$.

The runtime analysis is based on a non-negative potential function Φ. We partitioned up our iterations into three groups: shrinking, expanding, and neutral. Expanding and neutral iterations are bounded by $2n$ times the number of shrinking iterations that can occur. Furthermore, the initial value of Φ is 0 and shrinking iterations always decrease Φ by 1. This means that the number of shrinking iterations is bounded by Q, where Q is the maximum increase to Φ over all iterations.

Finally, we saw how we can initialize and terminate the algorithm. Initialization requires us to find a Δ^{start}-feasible pair (f, μ), and uses both cycle-cancelling and the Tight-Flow subroutine. The Tight-Flow subroutine simply finds a maximum flow into t using only the tight arcs. At termination, we call Tight-Flow again with our optimal μ^*. Computing a maximum flow into t with μ^* will bring us to the optimal flow f^*.

Notation	Meaning
Chapter 1: Introduction	
γ_{ij}	Gain factor of arc ij, always > 0
b_i	Demands (> 0) or supplies (< 0) at node i
e_i	Excess at node i, $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$
f_{ij}	Flow on arc ij
μ_i	Label at node i; inverse of dual solution, always > 0
E_f	The set of residual arcs: $\{ij : f_{ij} < u_{ij}\} \cup \{ji : f_{ij} > 0\}$
G_f	The residual graph with node set V and arc set E_f
Introduced in Chapter 2: Definitions and Notations	
f^μ_{ij}, b^μ_i, e^μ_i, γ^μ_{ij}	Relabelled quantities; see Subsection 2.2.2
$E^\mu_f(\Delta)$	The set of Δ-fat arcs: $E \cup \{ji : f^\mu_{ij} > \Delta\}$
$G^\mu_f(\Delta)$	The graph with node set V and arc set $E^\mu_f(\Delta)$
R_i	The reserve at i; $\sum_{ji:\gamma^\mu_{ji}<1} \gamma_{ji} f_{ji}$
T_0	High excess nodes; $T_0 \subseteq \{i : e^\mu_i \geq (\deg(i) + 2)\Delta\}$
T	Reachable nodes; $T \subseteq \{i : i \text{ is reachable from a node in } T_0 \text{ via a tight path in } E^\mu_f(\Delta)\}$
L	Low excess nodes; $\{i : e^\mu_i < (\deg(i) + 1)\Delta\}$

Notation	Meaning
Introduced in Chapter 3: Weakly Polynomial Algorithm	
α_1	Smallest scaling factor needed to tighten an arc in $E[T, V\backslash T]$
α_2	Smallest scaling factor needed to raise $e_i'^{\mu'}$ to $4(\deg(i)+2)\Delta'$ for $i \in V\backslash T$
Φ	Potential function for measuring number of iterations
Q	Total increase to Φ over all iterations

Table 3.2: Summary of Notations in Chapter 3

Chapter 4

Strongly Polynomial Algorithm

It is well known that if we use a scaling algorithm and there exists an arc ij such that the flow f_{ij} is large relative to Δ, then there exists an optimal solution where ${f^*}_{ij} > 0$. Call such an arc an abundant arc. This tells us that there exists optimal labels μ^* such that $\gamma_{ij}^{\mu^*} = 1$ (see Section 2.1), and we can contract the arc without losing information about the dual solution. This is the idea behind the strongly polynomial algorithm.

We will first see how abundant arcs are defined in generalized flows, and how they should be contracted. Next, we want to find conditions that can guarantee the appearance of abundant arcs in a strongly polynomial number of iterations. This is done through the new subroutine Filtration. Finally, the chapter will conclude by giving an overview of the runtime analysis.

Végh called the set of iterations between two contractions a major iteration. We will look at the algorithm's structure and its runtime in a major iteration.

A table of all the notations up to Chapter 4 is provided on page 71 at the end of the chapter. This will be useful for readers who are interested in a particular section rather than the full thesis.

4.1 Abundant Arcs

4.1.1 Definition of Abundant Arcs

For an arc ij, suppose we can guarantee that there exists an optimal flow f^* such that ${f^*}_{ij} > 0$. By the second complementary slackness condition (see Section 2.1), there exists optimal labels μ^* such that $\mu_j^* = \gamma_{ij}\mu_i^*$. Since our algorithm is seeking optimal labels rather than an optimal flow, we can contract the arc ij without losing any information. Optimal labels on the contracted network can be mapped back to optimal labels on the original network, and an optimal flow can then be computed.

Lemma 19. *If $f_{ij}^{\mu}/\Delta \geq 17m$, then there exists an optimal flow f^* where ${f^*}_{ij} > 0$. We will call ij an abundant arc.*

To prove Lemma 19, we will use the next lemma to bound the maximum change in flow on arcs between the current solution and the optimal solution. The proof for Lemma 20 will be deferred until Chapter 5.2.

Lemma 20. *Given a current flow f, labels μ, and scaling parameter Δ, there exists an optimal flow f^* and labels μ^* such that*

$$\max_{ij\in E} |f_{ij}^{\mu} - {f^*}_{ij}^{\mu^*}| \leq \sum_{i\in V-t} e_i^{\mu}(f) + (m+1)\Delta$$

Proof of Lemma 19. Using Lemma 20, we know that:

$$\begin{aligned}\max_{ij\in E} |f_{ij}^{\mu} - {f^*}_{ij}^{\mu^*}| &\leq \sum_{i\in V-t} e_i^{\mu}(f) + (m+1)\Delta \\ &\leq \sum_{i\neq t} 4(\deg(i)+2)\Delta + (m+1)\Delta \\ &\leq (8m+8n-8)\Delta + (m+1)\Delta < 17m\Delta\end{aligned}$$

Thus if $f_{ij}^{\mu} \geq 17m\Delta$, then ij has positive flow in some optimal solution. □

To understand how contractions work, it is simpler to consider our relabelled network. If pq is an abundant arc, then pq must be tight because (f,μ) is Δ-feasible. Assume that $p \neq t$. Let k be the new node obtained by contracting pq.

For the new node k, we will set:

- $\mu_k' = \mu_q$
- $b_k^{\mu} = b_p^{\mu} + b_q^{\mu}$

For the images of the original arcs ij (i.e. $ij \neq pq$) and original nodes i (i.e. $i \neq k$) on the contracted network, we will set:

- $\gamma^{\mu}_{\text{image}(ij)} = \gamma^{\mu}_{ij}$
- $\mu_{\text{image}(i)} = \mu_i$
- $b^{\mu}_{\text{image}(i)} = b^{\mu}_{i}$

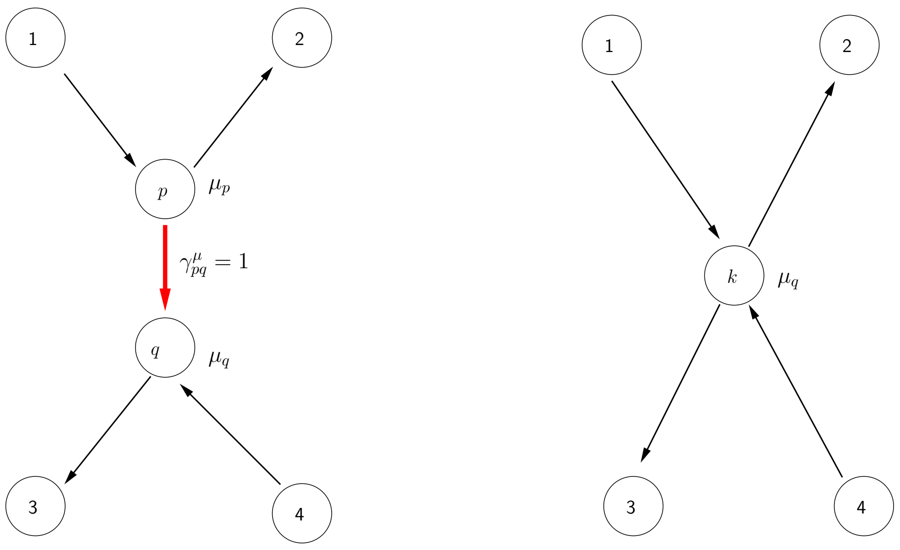


Figure 4.1: How to contract an arc

To find the real gain factors and supplies/demands for the contracted network, we can reverse the definitions of the relabelled terms. If $p = t$, then the new sink continues to get the label of 1 rather than μ_j, and the rest of the transformation follows.

For simpler presentation of the algorithm, we can reset our flow f and the scaling parameter Δ using Initialization.

We can add a step (lines 15-16) to contract arcs to our weakly polynomial algorithm in Subsection 3.1.1, and replace the terminating condition with $|V| = 1$.

```
Algorithm 4: Main Algorithm with Contractions
1  Initialization: Find a $\Delta^{\text{start}}$-feasible pair $(f, \mu)$ using Tight-Flow, and set $T_0 = T = \emptyset$;
2  while $|V| > 1$ do
3      while $\nexists ij : f^{\mu}_{ij}/\Delta \geq 17m$ do
4          Build $L = \{i : e^{\mu}_i < (\deg(i) + 1)\Delta\}$;
5          if $T \cap L \neq \emptyset$ then
6              Send $\Delta$ units of relabelled excess from $i \in T_0$ to $j \in L$ on a tight $i$-$j$ path
               in $E^{\mu}_f(\Delta)$;
7              if $e^{\mu}_i < (\deg(i) + 2)\Delta$ then remove $i$ from $T_0$;
8              Reset $T = T_0$;
9          else if $\exists$ tight arc $ij$, such that $ij \in E[T, V \backslash T]$ then
10             $T = T \cup \{j\}$;
11         else
12             Proceed to Elementary Step;
13         end
14     end
15     Find all arc $ij$ with $f^{\mu}_{ij}/\Delta \geq 17m$. Contract these arcs;
16     Reset $f$ and $\Delta$ using Initialization;
17 end
18 Termination: Expand network. Perform Tight-Flow using optimal $\mu^*$;
```

4.1.2 Guaranteeing Abundant Arcs

The fact that abundant arcs can be contracted is a well-known property. Rather, the challenge is to guarantee the existence of abundant arcs in our network without observing the value of flows on arcs. We use information about the supplies/demands, labels, and scaling parameter:

Lemma 21. *If $|b^{\mu}_i|/\Delta \geq 20mn$ for $i \in V - t$, then there is an abundant arc going in or out of node i.*

Proof. Assume by contradiction that we do not have an abundant arc. We will look at the supply and demand case separately. By feasibility of μ, we know that $\gamma^{\mu}_{ij} \leq 1$ for all arcs.

Case 1: i is a demand node
We can obtain a contradiction using $e_i^\mu \geq 0$ and $b_i^\mu \geq 20mn\Delta$.

$$0 \leq e_i^\mu = \sum_{j:ji\in E} \gamma_{ji}^\mu f_{ji}^\mu - \sum_{j:ij\in E} f_{ij}^\mu - b_i^\mu \leq (17m\Delta)(\deg(i)) - 0 - 20mn\Delta < 0$$

Case 2: i is a supply node
We can obtain a contradiction using $e_i^\mu \leq 4(\deg(i)+2)\Delta$ and $b_i^\mu \leq -20mn\Delta$.

$$\begin{aligned}(4n+8)\Delta \geq 4(\deg(i)+2)\Delta \geq e_i^\mu &= \sum_{j:ji\in E} \gamma_{ji}^\mu f_{ji}^\mu - \sum_{j:ij\in E} f_{ij}^\mu - b_i^\mu \\ &\geq 0 - (17m\Delta)(\deg(i)) + 20mn\Delta \\ &\geq -17m(n-1)\Delta + 20mn\Delta \\ &\geq (3mn + 17m)\Delta\end{aligned}$$

□

One of the benefit of using $|b_i^\mu|/\Delta = |b_i|/(\mu_i\Delta)$ is that this term only changes under very specific conditions. We know $|b_i|$ is a constant, and μ_i and Δ only change in the Elementary Step. Elementary Step updates $\Delta' = \Delta/\alpha$ and gives us two scenarios:

- If $i \in T$, then $\mu_i' = \alpha\mu_i$. Thus $|b_i^{\mu'}|/\Delta' = |b_i^\mu|/\Delta$.
- If $i \in V\backslash T$, then $\mu_i' = \mu_i$. Thus $|b_i^{\mu'}|/\Delta' = \alpha|b_i^\mu|/\Delta$.

As $\alpha > 1$, this means $|b_i^\mu|/\Delta$ is monotone increasing and can only get closer to being incident to an abundant arc. The increases only occur when $i \in V\backslash T$. Furthermore, the current value of $|b_i^\mu|/\Delta$ could be extremely small such that it becomes impossible to measure the progress until the $20mn$ threshold can be met.

As a result, we want to focus on nodes where $|b_i^\mu|/\Delta$ is greater than a lower threshold that is "comparable to n", so that we know how much progress must be made before it is guaranteed to be incident to an abundant arc. We take this lower threshold to be $1/n$. We will define a set of nodes D that are close to contractions because their $|b_i^\mu|/\Delta$ values are above $1/n$ (see Figure 4.2). Then:

$$D = \{i \in V - t : \frac{|b_i^\mu|}{\Delta} \geq 1/n\}$$

Since the values of $|b_i^\mu|/\Delta$ are monotone increasing, nodes that entered D cannot leave D within a major iteration. Furthermore, if $(V \backslash T) \cap D \neq \emptyset$, then running Elementary Step would provide measurable progress towards the next contraction because we are increasing some $|b_i^\mu|/\Delta$ that is reasonably large.

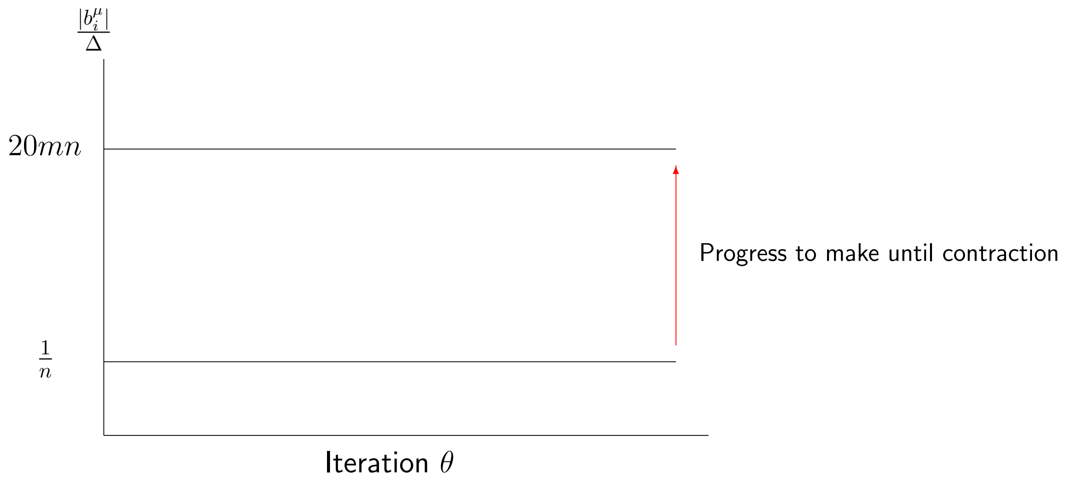

$\frac{|b_i^\mu|}{\Delta}$
$20mn$
Progress to make until contraction
$\frac{1}{n}$
Iteration θ

Figure 4.2: Bounded progress until contraction when nodes are in D

Clearly, a problem occurs when we run Elementary Step with $(V \backslash T) \cap D = \emptyset$. This brings us to the subroutine Filtration, which will fix our network to ensure that Elementary Step is making some progress even when $(V \backslash T) \cap D = \emptyset$.

4.2 Filtration

The Filtration subroutine is called whenever we need to proceed to Elementary Step but $(V \backslash T) \cap D = \emptyset$.

4.2.1 Purpose of Filtration and Tight-Flow

The most desirable outcome would be to add nodes in $V \backslash T$ to D. For example, if the excesses of nodes in $V \backslash T$ are extremely small and Elementary Step will choose a large α, then we might be able to push some $|b_i^\mu|/(\Delta/\alpha)$ over the $1/n$ threshold and add i to D.

We will attempt to do this by running a modified version Tight-Flow on $V \backslash T$ to move flow to the sink. Filtration will then update the current flow f with the flow on $V \backslash T$ computed by Tight-Flow.

Filtration's version of Tight-Flow on $V \backslash T$ runs in a similar manner to Tight-Flow on V as described in Subsection 3.4.1. We will look at the subgraph on $V \backslash T$ containing only the tight arcs. That is, the arc set is $E_\mu^{\text{Tight}}[V \backslash T] = \{ij \in E[V \backslash T] : \gamma_{ij}^\mu = 1\}$. Nodes will get supplies and demands b_i^μ. On this subgraph, we want to maximize the flow into t while maintaining non-negative excess at all other nodes. It is not clear that there is a feasible solution. In fact, how do we know that $V \backslash T$ contains supply nodes to satisfy the demands within this subset of nodes? Claim 22 (below) proves that a feasible solution exists.

Let us assume for now that Tight-Flow on $(V \backslash T, \mu)$ has a feasible solution. Tight-Flow computes a flow g that sends as much excess from the supplies within $V \backslash T$ to the sink t as possible, using only tight arcs. Since relabelling the subgraph and using only tight arcs means that we have scaled everything down to the same unit of measure, we should expect that the excesses are bound by the supplies and demands on the subgraph. We will see the formal statement of the lemmas and their proofs after the algorithm:

Algorithm 5: Filtration

```
1  Tight-Flow on $(V \backslash T, \mu)$: begin
2      Construct subgraph with nodes $V \backslash T$;
3      Add all tight arcs with both ends in $V \backslash T$ (i.e. $E_\mu^{\text{Tight}}[V \backslash T]$);
4      All arcs get capacities $u_{ij} = \infty$;
5      All nodes except $t$ gets supplies/demands $b_i^\mu$. The sink $t$ gets demand of 0;
6      Compute $g$: A maximum flow into $t$ such that all nodes have non-negative
       excesses;
7  end
8  for $ij \in E$ do
9      if $ij \in E[T] \cup E[T, V \backslash T]$ then $f_{ij}^{\prime\mu} = f_{ij}^\mu$;
10     if $ij \in E[V \backslash T, T]$ then $f_{ij}^{\prime\mu} = 0$;
11     if $ij \in E[V \backslash T]$ then $f_{ij}^{\prime\mu} = g_{ij}$;
12 end
```

Lines 8-12 are a new flow f' on our original network, based on the flow g computed by Tight-Flow.

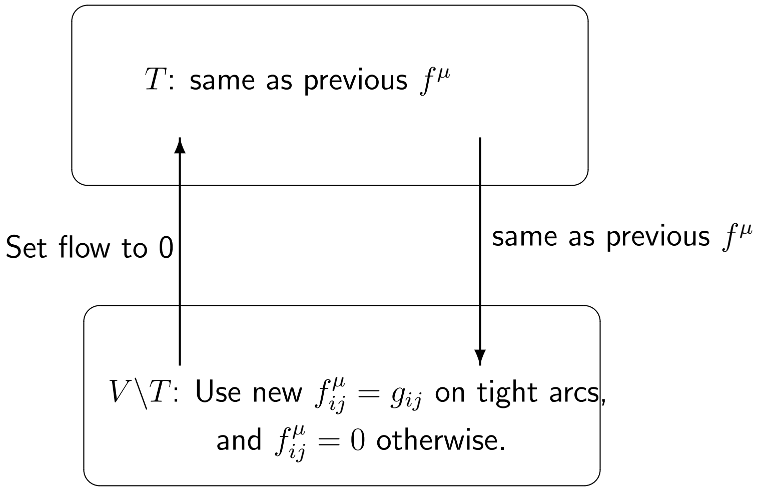

T: same as previous f^μ
Set flow to 0
same as previous f^μ
$V\backslash T$: Use new $f^\mu_{ij} = g_{ij}$ on tight arcs,
and $f^\mu_{ij} = 0$ otherwise.

Figure 4.3: Flow update in Filtration

For this chapter only, I will use the notation $e^\mu_i(g) = \sum_{ji \in E^{\text{Tight}}_\mu[V\backslash T]} g_{ji} - \sum_{ij \in E^{\text{Tight}}_\mu[V\backslash T]} g_{ij} - b^\mu_i$ for $i \in V\backslash T - t$ (where b^μ_i refers to to relabelled demand/supply on the original network).

Claim 22. *In Filtration, the Tight-Flow computation on $(V\backslash T, \mu)$ has a feasible solution g, such that $e^\mu_i(g) \geq 0$ for $i \in V\backslash T - t$.*

Proof. Consider the current flow f. There are no tight arcs in $E[T, V\backslash T]$, otherwise we should add a node to T rather than proceed to Filtration. By the definition of Δ-feasibility (see Definition 10), we can remove all the flow on incoming non-tight arcs because $e_i \geq R_i$. We can also remove flow on outgoing arcs without reducing excesses (see Figure 4.4). Thus, our current relabelled flow restricted to the subgraph on $V\backslash T$ with tight arcs is a feasible solution.

By letting $x_{ij} = f^\mu_{ij}$ on tight arcs in $E[V\backslash T]$, the relabelled flow becomes a feasible flow in this subgraph. □

Lemma 23. *The Tight-Flow computation will return a flow g on the subgraph such that $e^\mu_i(g) \leq n \max_{j \in V\backslash T - t} |b^\mu_j|$ for $i \in V\backslash T - t$.*

Corollary 24. *The Filtration subroutine will return a flow f' such that $e^\mu_i(f') \leq R'^\mu_i + n \max_{j \in V\backslash T - t} |b^\mu_j|$ for $i \in V\backslash T - t$.*

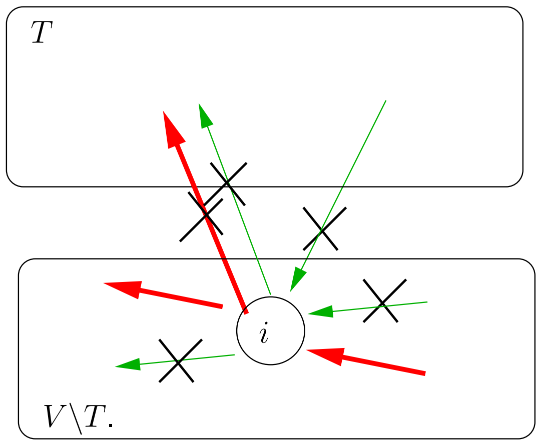


Figure 4.4: Feasibility in Tight-Flow $(V\backslash T, \mu)$. Crossed out arcs do not ruin the feasibility of the flow.

The corollary follows because all the flow on $E[V\backslash T, T]$ have been set to 0 and does not affect the excess. Consider a node $i \in V\backslash T - t$. When we map g back to the full network and define f', the relabelled excess of node i is $e_i^\mu(g)$ plus the flow on $ji \in E[T, V\backslash T]$. Since all arcs in $E[T, V\backslash T]$ non-tight, they contribute equally to excesses $e_i^\mu(f')$ and $R_i'^\mu$ (see Definition 9). We will now see the proof for Lemma 23.

Proof. This lemma clearly holds for nodes with zero excesses. Let us consider a particular node k where $e_k^\mu(g) > 0$. As Tight-Flow is a maximum flow computation, we can consider the standard residual graph of the flow g. Let Z be the set of nodes reachable from k using arcs in the residual graph of g. We know $t \notin Z$, otherwise we can increase the flow at t by pushing some ϵ amount from k. Following the same argument about Z as in Lemma 18, we know there is no arc ij with $i \in Z$, $j \notin Z$. Moreover, all original arc ji entering Z has zero flow, otherwise we have a residual arc ij leaving Z.

$$\begin{aligned}
e_k^\mu(g) &\leq \sum_{i\in Z} e_i^\mu(g)\\
&= \sum_{i\in Z}\left(\sum_{ji\in E_\mu^{\text{Tight}}[V\backslash T]} g_{ji} - \sum_{ij\in E_\mu^{\text{Tight}}[V\backslash T]} g_{ij} - b_i^\mu\right)\\
&= \sum_{i\in Z}\left(\sum_{\substack{ji\in E_\mu^{\text{Tight}}[V\backslash T]:\\ j\in Z}} g_{ji} - \sum_{\substack{ij\in E_\mu^{\text{Tight}}[V\backslash T]:\\ j\in Z}} g_{ij}\right) - \sum_{i\in Z} b_i^\mu\\
&\leq \sum_{i\in Z} |b_i^\mu|\\
&\leq n \max_{j\in V\backslash T-t} |b_j^\mu|
\end{aligned}$$

Since we only sum over the flow on arcs with both ends in Z, the excesses in Z are created only from the supplies in Z.

□

Running the Filtration subroutine could mean that we are no longer eligible to call Elementary Step. Recall from Subsection 3.2.2 that in order to run Elementary Step, we must have no low excess nodes in T (i.e. $T \cap L = \emptyset$) and all nodes in T_0 must satisfy the definition of being a high excess node. When we replace the flows on $E[V\backslash T, T]$ with zero flow, we could have reduced the excess of a node in T so that it violates one of these conditions. Consequently, we should update our sets and move onto the next iteration without performing Elementary Step. The strongly polynomial algorithm is presented next:

Algorithm 6: Main Algorithm, strongly polynomial version

```
1  Initialization: Find a $\Delta^{\text{start}}$-feasible pair $(f, \mu)$ using Tight-Flow, and set $T_0 = T = \emptyset$;
2  while $|V| > 1$ do
3      while $\nexists ij : f^{\mu}_{ij}/\Delta \geq 17m$ do
4          Build $L = \{i : e^{\mu}_i < (\deg(i) + 1)\Delta\}$;
5          if $T \cap L \neq \emptyset$ then
6              Send $\Delta$ units of relabelled excess from $i \in T_0$ to $j \in L$ on a tight $i$-$j$ path
               in $E^{\mu}_f(\Delta)$;
7              if $e^{\mu}_i < (\deg(i) + 2)\Delta$ then remove $i$ from $T_0$;
8              Reset $T = T_0$;
9          else if $\exists$ tight arc $ij$, such that $ij \in E[T, V \backslash T]$ then
10             $T = T \cup \{j\}$;
11         else
12             if $(V \backslash T) \cap D = \emptyset$ then Call Filtration;
13             if $\exists i \in T_0 : e^{\mu}_i < (\deg(i) + 2)\Delta$ then
14                 Remove $i$ from $T_0$ and reset $T = T_0$;
15                 Proceed to next iteration (go to line 3);
16             if $\exists i \in T : e^{\mu}_i < (\deg(i) + 1)\Delta$ then Proceed to next iteration (go to line
               3);
17             Call Elementary Step;
18         end
19     end
20     Find all arc $ij$ with $f^{\mu}_{ij}/\Delta \geq 17m$. Contract these arcs;
21     Reset $f$ and $\Delta$ using Initialization.
22 end
23 Termination: Expand network. Perform Tight-Flow using optimal $\mu^*$;
```

Lines 11-17 are where the changes occur. Line 12 checks whether we should run Filtration. Lines 12 to 15 checks whether we are allowed to proceed to Elementary Step after the flow update from Filtration.

4.2.2 Analysis of Filtration

Let assume that we proceed to Elementary Step after Filtration. If we skip Elementary Step, then T has shrunk (step 12-14) or T will shrink in the next iteration because $T \cap L \neq \emptyset$ and we will send flow (step 15).

Running Filtration before Elementary Step gives us the following benefits:

1. This is the only operation that sends flow directly to the sink t through the Tight-Flow computation.
2. If we proceed to Elementary Step, then our α will either add a node in $V \backslash T$ to D or tighten an arc going into a low excess node.

When we consider the weakly polynomial algorithm, we were always sending flow to some low excess node, which may or may not be t. The total excess over all $i \in V - t$ may not necessarily have decreased, and could simply have been rebalanced. On the other hand, when we run Tight-Flow in Filtration, excesses are sent directly to the sink.

The second benefit allows us to make faster progress. Previously, Elementary Step may either add a node to T_0 or tighten an arc (see Section 3.2). After running Elementary Step, it is possible that there is still no tight paths from T_0 to L. For example, we could have tightened ij where $j \notin L$. If we run Elementary Step after Filtration, we are guaranteed to either add nodes to D, which can happen at most $n-1$ times, or we will tighten ij with $j \in L$.

Claim 25. *If we proceed to Elementary Step after Filtration, then the α chosen will either add a node in $V \backslash T$ to D or tighten an arc going into a low excess node.*

Proof. Assume that we do not tighten an arc going into a low excess node. Let e_i, Δ, μ_i denote the parameters before Filtration and e'_i, Δ', μ'_i denote the parameters after Elementary Step. We know that Elementary Step does not modify the flow again if we performed Filtration, because all arcs that have their flows modified by α were replaced with zero flows in Filtration.

Case 1: $\alpha = \alpha_1$ so that we tighten an arc $ji \in E[T, V \backslash T]$ (Here, ji refers to an original arc).

If $i \in L$, then our claim is true. So assume $e'_i/\mu'_i \geq (\deg(i)+1)\Delta'$. Since $\mu'_i = \mu_i$, running Filtration before Elementary Step means that (the inequalities are explained below):

$$\begin{aligned}(\deg(i)+1)\Delta' \leq e'_i/\mu'_i &\leq R'^{\mu}_i + n \max_{k \in V \backslash T - t} |b^{\mu}_k| \\ &\leq \deg(i)\Delta' + n \max_{k \in V \backslash T - t} |b^{\mu}_k|\end{aligned}$$

The first line results from Corollary 24, and the second line comes from bounding the relabelled reserves (see discussion after Definition 10). By rearranging the last inequality, we get:

$$1/n \leq \frac{\max_{k \in V \backslash T - t} |b_k^{\mu}|}{\Delta'}$$

This means that a node $k \in V\backslash T - t$ is added to D, whereas it was not in D at the start of the iteration because we proceeded to Filtration.

Case 2: $\alpha = \alpha_2$ so that some node $i \in V\backslash T - t$ is added to T_0.

This means $e'_i/\mu'_i = 4(\deg(i)+2)\Delta'$. Since $e'_i/\mu'_i > (\deg(i)+1)\Delta'$, apply the same argument as in case 1. Again we added some $k \in V\backslash T - t$ to D. □

This claim, together with the two scenarios where we might skip Elementary Step, implies that we either expand D or shrink T within the next two iterations. Expanding D is desirable because it leads to more contractions. Furthermore, there are at most $n-1$ such iterations. Shrinking T also indicates good progress. As we showed in the weakly polynomial algorithm (see Section 3.3), the total number of shrinking iterations is bounded by $2nQ$, where Q is the total increase to our potential function. An overview of the different actions that the algorithm can take is summarized in the following diagram.

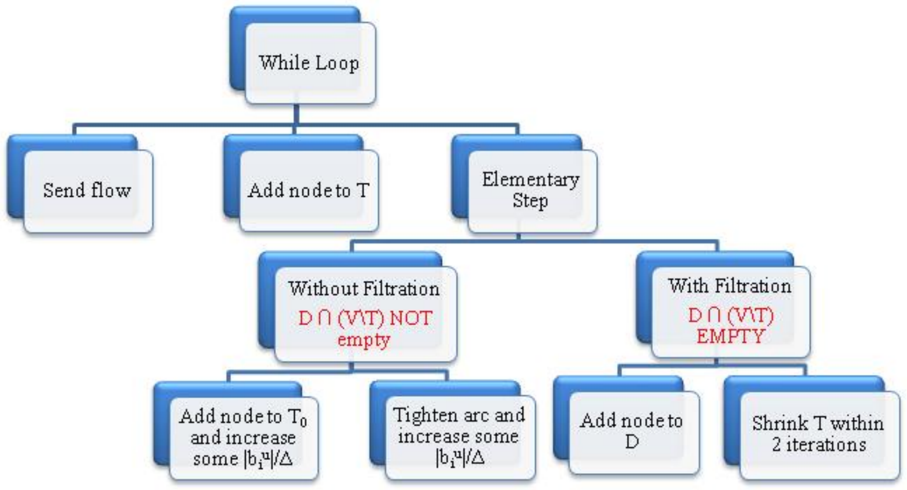

While Loop
Send flow
Add node to T
Elementary Step
Without Filtration
D ∩ (V\T) NOT empty
With Filtration
D ∩ (V\T) EMPTY
Add node to T_0 and increase some $|b_i^\mu|/\Delta$
Tighten arc and increase some $|b_i^\mu|/\Delta$
Add node to D
Shrink T within 2 iterations

Figure 4.5: Possible actions in each iteration

4.3 Runtime Analysis

We analyze the runtime for any one major iteration. Recall that a major iteration is the set of iterations between two contractions.The overall runtime is n times this quantity. Similar to the runtime analysis performed for the weakly polynomial algorithm, we will consider the increases and decreases to Φ (see Section 3.3) in a major iteration.

Recall from Section 3.3 that our potential function is $\Phi = \sum_{i \in T_0} \left(\left\lfloor \frac{e_i}{\mu_i \Delta} \right\rfloor - (\deg_i + 1) \right)$. Let us first consider the actions that could reduce the value of Φ. In the weakly polynomial algorithm, shrinking iterations consisted of flow augmentations and Elementary Steps where nodes were removed from T_0. These iterations decreased Φ by 1. In the strongly polynomial algorithm this continues to occur. In addition, we can now shrink T by running Filtration and removing nodes from T_0 rather than moving onto Elementary Step. Like the other shrinking iterations, this also reduces the value of Φ by 1. As a result, the number of shrinking iterations continue to be bounded by the total increase to Φ, denoted Q. Recall from Section 3.3 that $Q = \sum_{\text{iterations } \theta} \max\{\Phi^{(\theta+1)} - \Phi^{(\theta)}, 0\}$.

Expanding iterations do not change and occur for the same reason as in the weakly polynomial algorithm. There is one new possibility for a neutral iteration. This happens when we run Filtration and find that $T \cap L \neq \emptyset$. However, this means that at the next iteration, we must send flow. Thus neutral iterations continue to be followed by an expanding or a shrinking iteration. The total number of iterations remains $2nQ$ and Lemma 16 still holds.

Next, we need to consider an upper bound on Q. We will prove an upper-bound of Q for a major iteration rather than over the full algorithm. After a contraction, T_0 is reset to $\emptyset$ and we can bound the increases to Φ afresh. The problem with the weakly polynomial algorithm was that we could only give a weakly polynomial bound on Q. It is clear that if Q can be bounded by a strongly polynomial term in Theorem 17, then the total number of iterations must be strongly polynomial.

The following lemma will be proven in Section 5.1.

Lemma 26. *In a major iteration, node i enters T_0 at most $10n \log n$ times.*

Using this strongly polynomial bound on the number of times that i can enter T_0 and the same structure as Theorem 17, we can conclude with the proof on the total number of iterations over the whole algorithm.

Theorem 27. *The strongly polynomial algorithm terminates in at most $260mn^3 \log n$ iterations.*

Proof. Each time i enters T_0, it increases Φ by $[4(\deg(i)+2)-(\deg(i)+1)]$. Between two contractions, Lemma 26 implies that:

$$Q \leq \sum_{i\in V-t} [4(\deg(i)+2)-(\deg(i)+1)](10n\log n) \leq (6m+7n)(10n\log n) \leq 130mn\log n$$

Given that there are at most n contractions and $2nQ$ iterations in each major iteration, there are at most $260mn^3\log n$ iterations over the full algorithm. □

Because each iteration can run in strongly polynomial time, the total running time is strongly polynomial. We give a loose bound on the runtime of the algorithm next. Végh found a faster runtime by maintaining the previous flow f after every contraction (see Subsection 4.4.2), modifying Elementary Step, and using special data structures.

Theorem 28. *The total runtime of the algorithm is at most $O(m^2n^4\log^2 n)$.*

Proof. Consider the actions that we may perform in an iteration. If we augment flow, it takes us $O(m)$ time to identify and update a path. If we expand T by identifying a tight arc $ij \in E[T, V\backslash T]$, it also takes us $O(m)$ time. In Elementary Step, we compute α_1 and α_2, of which the latter will be more difficult to compute as we need to look at n nodes and m arcs. This takes $O(mn)$ time. We also check whether nodes can be added to or removed from T_0, which takes $O(n)$ time.

The bottleneck of Filtration is computing a traditional maximum flow in Tight-Flow. We could have non-integer values when we run Tight-Flow, so we can use the Sleator-Tarjan maximum flow algorithm [8], which runs in $O(mn\log n)$ time. We need to translate Tight-Flow into a traditional maximum flow problem with a source node instead of supplies and demands. This can be done by creating a source node s and arcs si for each $i \in V\backslash T - t$. Arc si is given a lower arc capacity of $l_{ij} = -\infty$ and an upper arc capacity of $u_{ij} = -b_i^\mu$. A maximum flow g to this problem always has $g_{si} = u_{si}$ because $\{s\}$ is the minimum cut (all other arcs have $u_{ij} = \infty$). Thus if i is a supply node, the arc si will provide b_i^μ units of flow. If i is a demand node, the arc si will provide $-b_i^\mu$ units of flow and we will need a net flow of b_i^μ units on the other arcs in order to have a feasible flow.

Finally, Initialization can be performed in $O(m^2n\log^2 n)$ time, using Radzik's maximum-mean-gain-cycle-cancelling algorithm [6, 7] (see Subsection 1.5.2, Section 1.6) and a maximum flow computation when we run Tight-Flow. Termination is also a Tight-Flow computation. Neither of these routines are bottlenecks in the runtime analysis.

Thus each iteration has a runtime of $O(mn\log n)$. There are $O(mn^3\log n)$ iterations (Theorem 27), which gives a total runtime of $O(m^2n^4\log^2 n)$. □

4.4 Others Considerations

4.4.1 Initialization and Termination

Initialization of the strongly polynomial algorithm is exactly the same as the weakly polynomial algorithm (see Subsection 3.4.2).

However, we terminate with one node remaining. We need optimal labels to the initial network. Since we know the order of contraction, we can undo the contractions in the reverse order. We know that if an arc ij was contracted, then $\gamma_{ij}^{\mu^*} = 1$ must hold and we can compute labels of an additional node with every expansion of the network. When we have expanded the whole network, we have optimal labels μ^* and we can invoke Tight-Flow(V, μ^*) to find an optimal flow.

Lemma 29. *Suppose the labels μ^* are optimal. Then Tight-Flow(V, μ^*) returns an optimal flow f^*.*

Proof. Tight-Flow can return some maximum flow g. Define f' such that $f'^{\mu^*}_{ij} = g_{ij}$.

Also, there exists an optimal solution f^* that satisfies the complementary slackness conditions along with μ^*. Since ${f^*}_{ij} = 0$ when $\gamma_{ij}^{\mu^*} < 1$ by complementary slackness conditions, $f^{*\mu^*}$ is a feasible solution to the Tight-Flow computation. Looking at the net flow into the sink t:

$$\sum_{jt} \gamma_{ji}^{\mu^*} f'^{\mu^*}_{jt} - \sum_{tj} f'^{\mu^*}_{tj} = \sum_{jt} g_{jt} - \sum_{tj} g_{tj} \geq \sum_{jt} \gamma_{ji}^{\mu^*} {f^*}^{\mu^*}_{jt} - \sum_{tj} {f^*}^{\mu^*}_{tj} \geq \sum_{jt} \gamma_{ji}^{\mu^*} f'^{\mu^*}_{jt} - \sum_{tj} f'^{\mu^*}_{tj}$$

The first inequality results from $f^{*\mu^*}$ being a feasible flow and g being an optimal flow to Tight-Flow. The second inequality follows from f' being a feasible flow and f^* being an optimal flow to the generalized flow problem. □

4.4.2 Reducing Runtime by $\log n$ Factor

In Végh's original paper, he was able to achieve at most $O(mn^3)$ iterations, thus shaving off a $\log n$ factor. He does this by keeping the current flow f after every contraction. However, if we contract i and j, the relabelled excess at the new node k would be $e_i^\mu + e_j^\mu$ and possibly greater than the threshold $4(\deg_k + 2)\Delta$. In order to synthesize the same conditions as Initialization so that $e_i^\mu < (\deg(i) + 2)\Delta'$ for all $i \in V - t$, Végh set $\Delta' = 16\Delta$. Simply put, he backtracks some progress in terms of the scaling parameter to keep the current flow.

The biggest effect is on the definition of D (see Subsection 4.1.2). We want to maintain the same set D rather than lose nodes because Δ increased somewhat arbitrarily. Thus, we would need to redefine D as follows:

$$D = \{i : |b_i^\mu|/\Delta \geq \frac{1}{16^C n}\}$$

Here, C represents the total number of contractions that have occurred. Every time we decrease the left term by 16 due to a contraction, we also decrease the right term. Therefore, every node that was in D before a contraction stays in D after the contraction. The total number of iterations where D can grow is $2n - 1$, which is the number of original and new nodes.

The runtime analysis is constructed in a similar manner, with a few modification to account for the iterations where we perform a contraction.

4.5 Summary

In this chapter, I introduced the contractions of abundant arcs. An arc ij is abundant if it satisfies $f_{ij}^\mu \geq 17m\Delta$, because we can guarantee that there exists an optimal solution where ${f^*}_{ij} > 0$. Since we know the relationship between μ_i^* and μ_j^*, we can contract the arc ij to work on a smaller network.

The difficulty is in showing that an abundant arc must appear within a strongly polynomial number of iterations. We know that $|b_i^\mu|/\Delta \geq 20mn$ implies that i is incident to an abundant arc. Using our set $D = \{i : |b_i^\mu|/\Delta \geq 1/n\}$ and our new subroutine Filtration, we were able to bound the number of iterations between two contractions. Many of the relevant lemmas will be proved in Section 5.1.

Although I presented the algorithm by resetting the flow f after every contraction, Végh's paper performed other updates so that we can continue with the current flow. This saves him a factor of $\log n$ in the runtime analysis, but adds another level of complexity.

Notation	Meaning
Introduced in Chapter 1: Introduction	
γ_{ij}	Gain factor of arc ij, always > 0
b_i	Demands (> 0) or supplies (< 0) at node i
e_i	Excess at node i, $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$
f_{ij}	Flow on arc ij
μ_i	Label at node i; inverse of dual solution, always > 0
E_f	The set of residual arcs: $\{ij : f_{ij} < u_{ij}\} \cup \{ji : f_{ij} > 0\}$
G_f	The residual graph with node set V and arc set E_f
Introduced in Chapter 2: Definitions and Notations	
f^{μ}_{ij}, b^{μ}_i, e^{μ}_i, γ^{μ}_{ij}	Relabelled quantities; see Subsection 2.2.2
$E^{\mu}_f(\Delta)$	The set of Δ-fat arcs: $E \cup \{ji : f^{\mu}_{ij} > \Delta\}$
$G^{\mu}_f(\Delta)$	The graph with node set V and arc set $E^{\mu}_f(\Delta)$
R_i	The reserve at i; $\sum_{ji:\gamma^{\mu}_{ji}<1} \gamma_{ji} f_{ji}$
T_0	High excess nodes; $T_0 \subseteq \{i : e^{\mu}_i \geq (\deg(i) + 2)\Delta\}$
T	Reachable nodes; $T \subseteq \{i : i \text{ is reachable from a node in } T_0 \text{ via a tight path in } E^{\mu}_f(\Delta)\}$
L	Low excess nodes; $\{i : e^{\mu}_i < (\deg(i) + 1)\Delta\}$

Notation	Meaning
Introduced in Chapter 3: Weakly Polynomial Algorithm	
α_1	Smallest scaling factor needed to tighten an arc in $E[T, V\backslash T]$
α_2	Smallest scaling factor needed to raise $e_i'^{\mu'}$ to $4(\deg(i)+2)\Delta'$ for $i \in V\backslash T$
Φ	Potential function for measuring number of iterations
Q	Total increase to Φ over all iterations
Introduced in Chapter 4: Strongly Polynomial Algorithm	
D	$\{i : \lvert b_i^\mu \rvert/\Delta \geq 1/n\}$
$E_\mu^{\text{Tight}}[V\backslash T]$	$\{ij \in E[V\backslash T] : \gamma_{ij}^\mu = 1\}$

Table 4.1: Summary of Notations in Chapter 4

Chapter 5

Runtime Analysis of Strongly Polynomial Algorithm

This chapter will present the proofs that were omitted in Chapter 4. A table of all the notations up to Chapter 5 is provided on page 83 at the end of the chapter. This will be useful for readers who are interested in a particular section rather than the full thesis.

5.1 Increases to Φ in Strongly Polynomial Algorithm

Let Θ be the set of all iterations. In order to bound Q - the total increase to our potential function Φ in a major iteration (see Section 3.3) - we need to look at two types of iterations:

- Θ^F: The set of iterations where we run Filtration
- Θ^D: The set of iterations where we grow D by at least one node (see Subsection 4.1.2)

We will show that Q is at most $130mn \log n$. Similar to the weakly polynomial algorithm, Φ only increases when a node enters T_0 through the Elementary Step and $e_i^\mu = 4(\deg(i) + 2)\Delta$. Our goal is to show that the following lemma holds in a major iteration:

Lemma 26. *In a major iteration, node i enters T_0 at most $10n \log n$ times.*

This implies that there are at most $260mn^3 \log n$ iterations in the algorithm, as proven in Theorem 27.

The proof of Lemma 26 will follow the structure of this flowchart:

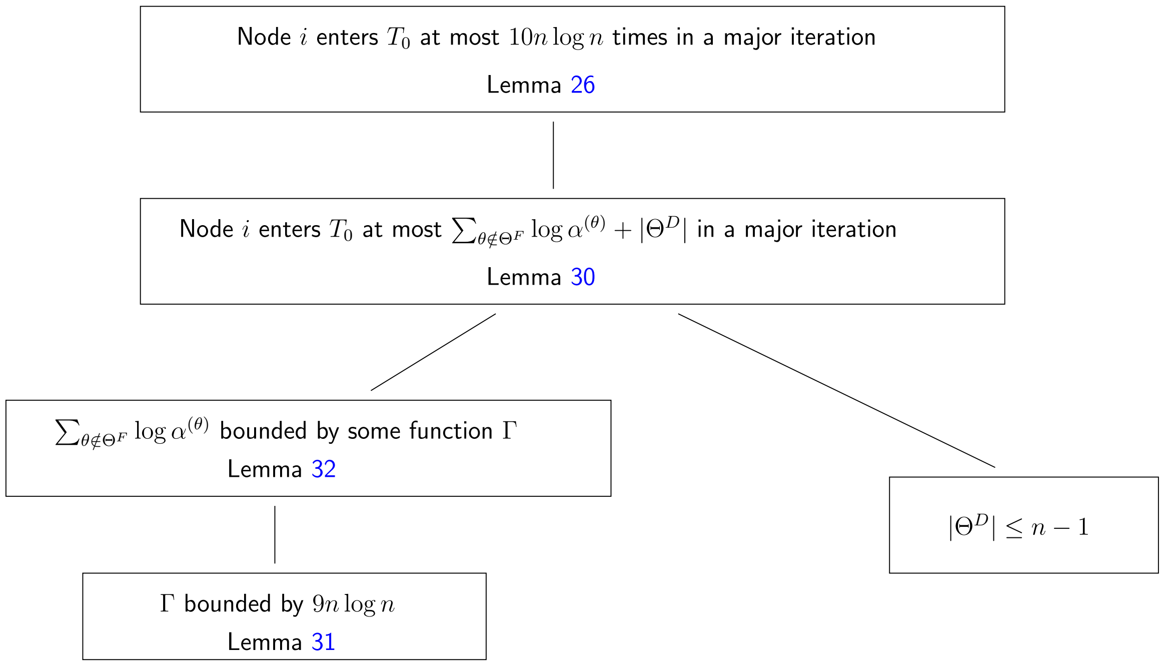

Node i enters T_0 at most $10n \log n$ times in a major iteration
Lemma 26
Node i enters T_0 at most $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)} + |\Theta^D|$ in a major iteration
Lemma 30
$\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)}$ bounded by some function Γ
Lemma 32
$|\Theta^D| \leq n - 1$
Γ bounded by $9n \log n$
Lemma 31

Figure 5.1: Flowchart for Lemma 26

We will always use the convention that $\alpha^{(\theta)} = 1$ if we do not perform Elementary Step in iteration θ.

Lemma 30. *In a major iteration, node i enters T_0 at most $\sum_{\theta \notin \Theta^F} \alpha^{(\theta)} + |\Theta^D|$ times.*

Proof. Within this proof, we will write e_i^μ as e_i/μ_i for the relabelled excess of a node i, because the proof could update one of e_i and μ_i and not the other in an iteration (see Table 5.1). In other words, the proof analyzes the relabelled excess over some updates to the labels μ.

Fix a node i that we will focus on throughout this proof. Let τ_1, τ_2, $\dots$, τ_λ be the iterations where node i enters T_0. For each $l = 1, \dots, \lambda$, there exists some iteration θ

between τ_{l-1} and τ_l such that:

$$\left(\frac{e_i}{\mu_i \Delta}\right)^{(\theta)} < (\deg(i) + 2)$$

The superscript (θ) indicates that we are looking at this term at the beginning of iteration θ. Let us consider why this statement is true. Before τ_1, the inequality would be true by the choice of Δ^{start}. Between iterations τ_{l-1} and τ_l for $l > 1$, i must have left T_0 after τ_{l-1}, at which time it would have satisfied the above inequality.

Within this section, we use the "interval notation" $[\tau_x, \tau_y]$ for $x \leq y$ to denote a set of consecutive iterations $\{\tau_x, \tau_x + 1, \ldots, \tau_y - 1, \tau_y\}$. Let $\bar{\tau}_l$ be the largest index before τ_l such that $(\frac{e_i}{\mu_i \Delta})^{(\bar{\tau}_l)} < (\deg(i) + 2)$. That is, for all iterations $\theta \in [\bar{\tau}_l + 1, \tau_l]$, we must have $(\frac{e_i}{\mu_i \Delta})^{(\theta)} \geq (\deg(i) + 2)$.

Case 1: $\exists \theta \in [\bar{\tau}_l, \tau_l]$ **such that** $\theta \in \Theta^D$
There are at most $n - 1$ such intervals in a major iteration.

Case 2: $\forall \theta \in [\bar{\tau}_l, \tau_l]$**,** $\theta \notin \Theta^D$
The next two sub-claims shows that we can give an upper-bound on the number of intervals as a function of α.

Sub-claim 1. *Let i be the specific node chosen at the start of the proof. If $i \in V \backslash T^{(\theta)}$, then we do not perform Filtration in iteration $\theta \in [\bar{\tau}_l, \tau_l - 1]$.*

Proof of subclaim 1. First, consider the case that we perform Filtration but not Elementary Step. Let e'_i be the excess at node i after Filtration. Then we can bound the new relabelled excess by:

$$\begin{aligned}\left(\frac{e_i}{\mu_i}\right)^{(\theta+1)} &= e_i'^{\mu} \leq R_i'^{\mu} + n \max_{j \in V \backslash T - t} |b_j^{\mu}| \\ &\leq \deg(i)\Delta + n\frac{\Delta}{n} \leq (\deg(i) + 1)\Delta\end{aligned}$$

The first inequality results from Corollary 24 because we do not modify μ when Elementary Step is not called. The bound on $R_i^{\mu'}$ follows from the discussion after Definition 10. The bound on $\max_{j \in V \backslash T - t} |b_j^{\mu}|$ results from $\theta \notin \Theta^D$, so that none of the nodes in $V \backslash T$ can enter D. This contradicts the choice of $\bar{\tau}_l$ since we must have $(e_i/(\mu_i \Delta))^{(\theta+1)} \geq (\deg(i) + 2)$ for $\theta \in [\bar{\tau}_l, \tau_l - 1]$.

Now consider the case where we perform Elementary Step after Filtration. The excess at i does not change after running Elementary Step, because the arcs that are updated (tight arcs in $E[V\backslash T]$ and all arcs in $E[V\backslash T, T]$) already have zero flow after running Filtration. In terms of excesses, labels, and scaling parameter:

Time	Excess	Label	Scaling Parameter
Beginning of iteration θ	e_i	μ_i	Δ
After Filtration	e'_i	μ_i	Δ
End of iteration θ = Beginning of iteration $\theta+1$	e'_i	$\mu'_i = \mu_i$	$\Delta' = \Delta/\alpha$

Table 5.1: Result of running Elementary Step after Filtration

Since $\theta + 1 \in [\bar{\tau}_l + 1, \tau_l]$, we know:

$$\begin{aligned}(\deg(i) + 2) \leq (\frac{e_i}{\mu_i \Delta})^{(\theta+1)} &= \frac{(e'_i/\mu'_i)}{\Delta'} \\ &\leq \frac{R_i'^{\mu'} + n \max_{j \in V \backslash T - t} |b_j^{\mu}|}{\Delta'} \\ &\leq \deg(i) + \frac{n \max_{j \in V \backslash T - t} |b_j^{\mu}|}{\Delta'}\end{aligned}$$

By rearranging, there exists j such that $|b_j^{\mu}|/\Delta' \geq \frac{2}{n}$. This is a contradiction to $\theta \notin \Theta^D$ because j can be added to D.

□

Sub-claim 2. *Let i be the specific node chosen at the start of the proof. If we run Elementary Step without Filtration in iteration θ, then $(\frac{e_i}{\mu_i \Delta})^{(\theta+1)} \leq (\alpha^{(\theta)})^2 \max\{(\frac{e_i}{\mu_i \Delta})^{(\theta)}, \deg(i)\}$.*

Proof of sub-claim 2. Let e'_i, μ'_i, Δ' be the parameters after running Elementary Step, which are equivalent to the parameters at the beginning of iteration $\theta + 1$ as we do not contract. Recall from Figure 3.4c that $F_3(i) = \{ij : \gamma_{ij}^{\mu} < 1, ij \in E[V\backslash T]\} \cup \{ij : ij \in E[V\backslash T, T]\}$, and all arcs in this set satisfies $f_{ij}^{\mu} \leq \Delta$. Elementary Step decreases the flow on arc $ij \in F_3(i)$ by $(1 - \frac{1}{\alpha})$. Then:

$$\begin{aligned}
\frac{e_i'}{\mu_i'\Delta'} = \frac{\alpha e_i'}{\mu_i\Delta} &\le \frac{\alpha}{\mu_i\Delta}\left[e_i + (1-\frac{1}{\alpha})\sum_{ij\in F_3(i)} f_{ij}\right] \\
&\le \frac{\alpha e_i}{\mu_i\Delta} + \frac{\alpha-1}{\Delta}\sum_{ij\in F_3(i)} f_{ij}^{\mu} \\
&\le \frac{\alpha e_i}{\mu_i\Delta} + \frac{\alpha-1}{\Delta}\deg(i)\Delta \\
&\le \alpha^2 \max\{\frac{e_i}{\mu_i\Delta}, \deg(i)\}
\end{aligned}$$

In the last inequality, we choose a loose bound of $\alpha^2 \ge 2\alpha - 1$ (recall $\alpha \ge 1$) because it will be more convenient in later proofs when we need to take logarithms. □

Returning to the proof of Lemma 30, we can prove the following facts for our chosen node i:

1. For iterations $\theta \in [\bar{\tau}_l + 1, \tau_l]$, we know $(e_i/(\mu_i\Delta))^{(\theta)} > \deg(i)$
2. For iterations $\theta \in [\bar{\tau}_l, \tau_l - 1]$, $\frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)},\deg(i)\}} \le (\alpha^{(\theta)})^2$ if $i \in V\backslash T^{(\theta)}$
3. For iterations $\theta \in [\bar{\tau}_l, \tau_l - 1]$, $\frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)},\deg(i)\}} \le 1$ if $i \in T^{(\theta)}$

Earlier, our choice of iteration $\bar{\tau}_l$ ensures that $(e_i/(\mu_i\Delta))^{(\theta)} > (\deg(i) + 2)$ for all iterations $\theta \in [\bar{\tau}_l + 1, \tau_l]$. Thus, Fact (1) must be true.

For Fact (2), we know that e_i^{μ} is not affected by augmentations because node $i \notin T$, nor is it affected by expansions of T. In these cases, $\mu_i'\Delta' = \mu_i\Delta$, so $\frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)},\deg(i)\}} \le \frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{(e_i/(\mu_i\Delta))^{(\theta)}} = 1 = (\alpha^{(\theta)})^2$. If we run Elementary Step, then Fact (2) is true by our two sub-claims.

For Fact (3), we know that node i is not sending flow because it is not in T_0. Furthermore, we know that node $i \notin L$, otherwise we would have $e_i'^{\mu'} < (\deg(i) + 1)\Delta' + \Delta' = (\deg(i)+2)\Delta'$ after the augmentation (see Section 2.4). So e_i^{μ} is unaffected by flow augmentations. If we expand T, there is no change to the relabelled excess. Both Filtration and Elementary Step will only reduce the excess into node i because we decrease the flow on arcs in $E[V\backslash T, T]$. Thus $e_i' \le e_i$ and $\mu_i'\Delta' = \mu_i\Delta$, so $\frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)},\deg(i)\}} \le \frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{(e_i/(\mu_i\Delta))^{(\theta)}} \le 1$.

For our chosen node i, let us consider the increase in $e_i/(\mu_i\Delta)$ from iteration $\bar{\tau}_l$ to iteration τ_l (see below for explanations).

$$\begin{aligned}
4 = \frac{4(\deg(i)+2)}{(\deg(i)+2)} &\leq \frac{(e_i/(\mu_i\Delta))^{(\tau_l)}}{\max\{(e_i/(\mu_i\Delta))^{(\bar{\tau}_l)}, \deg(i)\}} \\
&= \frac{(e_i/(\mu_i\Delta))^{(\bar{\tau}_l+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\bar{\tau}_l)}, \deg(i)\}} \times \prod_{\theta\in[\bar{\tau}_l+1,\tau_l-1]} \frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{(e_i/(\mu_i\Delta))^{(\theta)}} \\
&= \prod_{\theta\in[\bar{\tau}_l,\tau_l-1]} \frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)}, \deg(i)\}}
\end{aligned}$$

The first line is true because $(e_i/(\mu_i\Delta))^{(\tau_l)} = 4(\deg(i)+2)$ when i entered T_0. Furthermore, we chose $\bar{\tau}_l$ such that $\deg(i)+2 > \max\{(e_i/(\mu_i\Delta))^{(\bar{\tau}_l)}, \deg(i)\}$. In the second line, we use telescopic products to represent the previous term. In the third line, we know that $(e_i/(\mu_i\Delta))^{(\theta)} \geq (\deg(i)+2)$ for $\theta \in [\bar{\tau}_l+1, \tau-1]$, so $(e_i/(\mu_i\Delta))^{(\theta)} = \max\{(e_i/(\mu_i\Delta))^{(\theta)}, \deg(i)\}$ and we can combine all the terms.

Later, Claim 31 will help us bound $\alpha^{(\theta)}$ on iterations $\theta \notin \Theta^F$ (see Figure 5.1), so we want to focus on these iterations. We look at iterations where our chosen node i is in $V\backslash T$ because $\Theta^F \cap \{\theta : i \in V\backslash T^{(\theta)}, \theta \in [\bar{\tau}_l, \tau_l - 1]\} = \emptyset$ (by sub-claim 1). It is possible that our node i enters and leaves T throughout iterations $[\bar{\tau}_l, \tau_l - 1]$. That is, $i \in T^{(\theta')}$ and $i \in V\backslash T^{(\theta)}$ for some $\theta, \theta' \in [\bar{\tau}_l, \tau_l - 1]$ and $\theta \neq \theta'$. Using Fact 3, $e_i/(\mu_i\Delta)$ decreases when $i \in T^{(\theta')}$, and so we can ignore these iterations (second inequality below). Continuing our above analysis:

$$\begin{aligned}
4 &\leq \prod_{\theta\in[\bar{\tau}_l,\tau_l-1]} \frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)}, \deg(i)\}} \\
&\leq \prod_{\theta\in([\bar{\tau}_l,\tau_l-1]\cap\{\theta: i\in V\backslash T^{(\theta)}\})} \frac{(e_i/(\mu_i\Delta))^{(\theta+1)}}{\max\{(e_i/(\mu_i\Delta))^{(\theta)}, \deg(i)\}} \\
&\leq \prod_{\theta\in([\bar{\tau}_l,\tau_l-1]\cap\{\theta: i\in V\backslash T^{(\theta)}\})} (\alpha^{(\theta)})^2 \\
&\leq \prod_{\theta\in([\bar{\tau}_l,\tau_l-1]\backslash\Theta^F)} (\alpha^{(\theta)})^2
\end{aligned}$$

The third inequality follows from Fact 2. The last inequality follows from $\alpha^{(\theta)} \geq 1$ and

$([\bar{\tau}_l, \tau_l - 1] \cap \{\theta : i \in V \backslash T^{(\theta)}\}) \subseteq ([\bar{\tau}_l, \tau_l - 1] \backslash \Theta^F)$, because we cannot run Filtration when $i \in V \backslash T$.

We can take logarithms on both sides of the inequality.

$$1 \leq \sum_{\theta \in [\bar{\tau}_l, \tau_l - 1] \backslash \Theta^F} \log \alpha^{(\theta)} \leq \sum_{\theta \in [\tau_{l-1}+1, \tau_l] \backslash \Theta^F} \log \alpha^{(\theta)}$$

The second inequality is true because $\alpha^{(\theta)} \geq 1$ for all iterations θ, so we can sum over a larger interval. Finally, take the sum over all intervals where $[\tau_{l-1} + 1, \tau_l] \cap \Theta^D = \emptyset$. We also know there are at most $|\Theta^D|$ intervals where we can add a node into D, giving us:

$$\lambda \leq \sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)} + |\Theta^D|$$

□

It is clear that $|\Theta^D| \leq n - 1$, so the challenge is to bound the other term of the above inequality.

5.1.1 Bounding $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)}$

In order to find an upper bound on this term, we use a secondary potential function, Γ. Γ will be a non-negative function that increases when we add nodes to D but decreases by $\alpha^{(\theta)}$ in each non-filtration iteration θ. By this, we mean iterations $\theta \notin \Theta^F$. This will allow us to bound $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)}$. First, for each node $i \in D$, define:

$$\Gamma_i = \log \frac{32mn\Delta}{|b_i^\mu|}$$

Since the value of $|b_i^\mu|/\Delta = |b_i|/(\mu_i \Delta)$ is monotone increasing, we know that Γ_i is monotone decreasing. Let $\Gamma = \sum_{i \in D} \Gamma_i$ and consider how we can increase or decrease Γ. Recall from Subsection 4.1.2 that when a node enters D, it stays in D throughout the major iteration. The next two lemmas show that $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)} \leq$ Total increase to $\Gamma \leq 9n \log n$.

Claim 31. *In a major iteration, the total increase to Γ is at least $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)}$.*

Proof. We claim that Γ_i decreases by $\log \alpha$ for some node $i \in D$ in each non-filtration iteration. There are three actions that we can perform in an iteration $\theta \notin \Theta^F$:

- Augmenting flow - Γ_i remains the same for all nodes $i \in D$, and $\alpha = 1 \Rightarrow \log \alpha = 0$.
- Expanding T - Γ_i remains the same for all nodes $i \in D$, and $\alpha = 1 \Rightarrow \log \alpha = 0$.
- Elementary Step - Since we do not run Filtration, there is some node $i \in (V \backslash T) \cap D$. At the end of the iteration, $\Gamma_i' = \log \frac{32mn(\Delta/\alpha)}{|b_i^\mu|} = \Gamma_i - \log \alpha$.

Furthermore, we know that $\Gamma_i' > 0$ at the end of an iteration (except the last iteration of a major iteration), otherwise $32mn\Delta' \leq |b_i^\mu|$ implies that we can contract an arc incident to i. By counting the increases to Γ separately, we know that Γ decreases by at least $\log \alpha$ in every iteration where we do not run Filtration. Since Γ is always a non-negative function, the total increase to Γ is at least the total decrease to Γ, which is $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)}$. □

Now that we have an upper bound on $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)}$ based on Γ, let use determine an upper bound on the total increase to Γ.

Claim 32. *In a major iteration, the total increase to* Γ *is at most* $9n \log n$*.*

Proof. Since Γ_i is monotone decreasing once node i enters D, the potential function Γ can only increase when we add a new node i to D. Including nodes at the start of a major iteration is considered adding them at the first iteration.

At entry, we know that $|b_i^\mu|/\Delta \geq 1/n$. This gives us:

$$\Gamma_i \leq \log 32mn^2 \leq 5 + \log m + \log n^2 \leq 5 + 4 \log n \leq 9 \log n$$

Summing up over $n-1$ nodes, the total increase to Γ is at most $9n \log n$. □

Claim 31 and Claim 32 tells us that $\sum_{\theta \notin \Theta^F} \log \alpha^{(\theta)} \leq$ Total increase to $\Gamma \leq 9n \log n$. Following the flowchart 4.5, we see that each node enters T_0 at most $10n \log n$ between two contractions and we obtain Lemma 26.

5.2 Abundant Arcs

In this section, we will see a proof of Lemma 20. This was omitted in Subsection 4.1.1, where we used Lemma 20 to show that $f^{\mu}_{ij} \geq 17m\Delta$ implies arc ij is abundant (Lemma 19):

Lemma 20. *Given a current flow f, labels μ, and scaling parameter Δ, there exists an optimal flow f^* and labels μ^* such that*

$$\max_{ij \in E} |f^{\mu}_{ij} - {f^*}^{\mu^*}_{ij}| \leq \sum_{i \in V-t} e^{\mu}_i(f) + (m+1)\Delta$$

Proof. Let us define an auxiliary flow $\tilde{f}$ from our current f to simplify our analysis:

$$\tilde{f}_{ij} = \begin{cases} f_{ij} : & \text{if } ij \text{ is tight} \\ 0 : & \text{if } ij \text{ is not tight} \end{cases}$$

If we can show that $\max_{ij\in E} |\tilde{f}^{\mu}_{ij} - {f^*}^{\mu^*}_{ij}| \leq \sum_{i\in V-t} e^{\mu}_i(\tilde{f})$ for some optimal flow f^*, then the lemma must be true. This is due to the triangle inequality and $\sum_{i\in V-t} e^{\mu}_i(\tilde{f}) \leq \sum_{i\in V-t} e^{\mu}_i(f) + m\Delta$, because we removed at most Δ units of relabelled flow from each non-tight arc.

$$|f^{\mu}_{ij} - {f^*}^{\mu^*}_{ij}| \leq |f^{\mu}_{ij} - \tilde{f}^{\mu}_{ij}| + |\tilde{f}^{\mu}_{ij} - {f^*}^{\mu^*}_{ij}| \leq \Delta + (\sum_{i\in V-t} e^{\mu}_i(f) + m\Delta)$$

Out of all optimal f^*, we will choose the optimal flow that minimizes $\sum_{ij} |\tilde{f}_{ij} - {f^*}_{ij}|$.

Define a new subgraph H, where $V(H)$ is the original node set and $E(H) = \{ij : \tilde{f}_{ij} < {f^*}_{ij}\} \cup \{ji : \tilde{f}_{ij} > {f^*}_{ij}\}$. The flow on the arcs are define as:

$$h_{ij} = \begin{cases} {f^*}_{ij} - \tilde{f}_{ij} : & \text{if } ij \in E \text{ (forward arc)} \\ \gamma_{ji}(\tilde{f}_{ji} - {f^*}_{ji}) : & \text{if } ji \in E \text{ (backward arc)} \end{cases}$$

We can interpret h_{ij} as the difference in flow between $\tilde{f}_{ij}$ and ${f^*}_{ij}$.

Let $E_{\tilde{f}}$, E_{f^*} denote the arcs in the residual graphs of flows $\tilde{f}$ and f^* respectively. Observe that $E(H) \subseteq E_{\tilde{f}}$, since we included a forward arc whenever $\tilde{f}_{ij}$ is not at capacity ($\tilde{f}_{ij} < {f^*}_{ij}$) and a backward arc whenever $\tilde{f}_{ij}$ has positive flow ($\tilde{f}_{ij} > {f^*}_{ij} \geq 0$). By the same argument, $E(\overleftarrow{H}) \subseteq E_{f^*}$ where $\overleftarrow{H}$ denotes the subgraph with reverse arcs of $E(H)$. We will use this to show the next sub-claim.

Sub-claim 3. *The subgraph H is acyclic.*

Proof of sub-claim 3. By contradiction, assume that there exists a directed cycle C contained in H. For this proof only, I will use the notation $\gamma(C) = \prod_{ij \in C} \gamma_{ij}$. Since we have a feasible μ, we know that $\gamma(C) = \gamma^{\mu}(C) \leq 1$. But then the reverse cycle $\overleftarrow{C}$ must be contained in $\overleftarrow{H}$, and $1/\gamma(C) = \gamma(\overleftarrow{C}) = \gamma^{\mu^*}(\overleftarrow{C}) \leq 1$. This means that $\gamma(C) = 1$ and $\gamma_{ij}^{\mu^*} = 1$ for arcs $ij \in C$.

We can push flow around C without gaining or losing excesses. In the optimal f^*, let us push $\epsilon > 0$ units of relabelled flow around C, meaning that we set the new flow as follows:

$$f^{*\mu^*}_{ij} = \begin{cases} f^{*\mu^*}_{ij} - \epsilon : & \text{if } ij \in C \text{ i.e. } (\tilde{f}_{ij} < f^*{}_{ij}) \\ f^{*\mu^*}_{ij} + \epsilon : & \text{if } ji \in C \text{ i.e. } (\tilde{f}_{ij} > f^*{}_{ij}) \end{cases}$$

Because we used relabelled flows and units, this has the same outcome as pushing flow on a cycle of a traditional maximum flow problem. Excesses at the nodes are unchanged. The way we modified f^* means that we decreased $\sum_{ij} |\tilde{f}_{ij} - f^*{}_{ij}|$ without changing the flow into sink t and this is a contradiction to our choice of f^*. □

Thus H is acyclic and contains all the paths that excesses in $\tilde{f}$ takes to become the optimal f^*. We can decompose H into at most m paths which we can use to push excesses into the sink t. Furthermore, one unit of relabelled excess leaving from node i can contribute at most one unit of relabelled flow arriving at sink t, and only if we use a tight i-t path (see Subsection 2.2.2). Thus, the total relabelled excesses that will go into sink t is bounded by $\sum_{i \in V-t} e_i^{\mu}(\tilde{f})$, as required.

□

5.3 Summary

This chapter contains two major proofs to support the strongly polynomial algorithm

In the first proof, I showed how we can bound the number of times that any node can enter T_0 between two contractions. This helps us bound the potential function Φ and the total number of iterations in the strongly polynomial algorithm. The proof looks at the change in relabelled excess for $i \in V \backslash T$ after an Elementary Step. By bounding the change and using a secondary potential function Γ, we saw a strongly polynomial bound on the number of times that i enters T_0.

In the second proof, I showed that given a current flow f, labels μ, and scaling parameter Δ, there exists an optimal flow f^* and labels μ^* such that $\max_{ij \in E} |f^\mu_{ij} - f^{*\mu^*}_{ij}| \leq \sum_{i \in V-t} e^\mu_i(f) + (m+1)\Delta$. This proof was based on flow decomposition techniques.

Notation	Meaning
Introduced in Chapter 1: Introduction	
γ_{ij}	Gain factor of arc ij, always > 0
b_i	Demands (> 0) or supplies (< 0) at node i
e_i	Excess at node i, $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$
f_{ij}	Flow on arc ij
μ_i	Label at node i; inverse of dual solution, always > 0
E_f	The set of residual arcs: $\{ij : f_{ij} < u_{ij}\} \cup \{ji : f_{ij} > 0\}$
G_f	The residual graph with node set V and arc set E_f
Introduced in Chapter 2: Definitions and Notations	
f^μ_{ij}, b^μ_i, e^μ_i, γ^μ_{ij}	Relabelled quantities; see Subsection 2.2.2
$E^\mu_f(\Delta)$	The set of Δ-fat arcs: $E \cup \{ji : f^\mu_{ij} > \Delta\}$
$G^\mu_f(\Delta)$	The graph with node set V and arc set $E^\mu_f(\Delta)$
R_i	The reserve at i; $\sum_{ji:\gamma^\mu_{ji}<1} \gamma_{ji} f_{ji}$
T_0	High excess nodes; $T_0 \subseteq \{i : e^\mu_i \geq (\deg(i)+2)\Delta\}$
T	Reachable nodes; $T \subseteq \{i : i$ is reachable from a node in T_0 via a tight path in $E^\mu_f(\Delta)\}$
L	Low excess nodes; $\{i : e^\mu_i < (\deg(i)+1)\Delta\}$

Notation	Meaning
Introduced in Chapter 3: Weakly Polynomial Algorithm	
α_1	Smallest scaling factor needed to tighten an arc in $E[T, V\backslash T]$
α_2	Smallest scaling factor needed to raise $e_i'^{\mu'}$ to $4(\deg(i)+2)\Delta'$ for $i \in V\backslash T$
Φ	Potential function for measuring number of iterations
Q	Total increase to Φ over all iterations
Introduced in Chapter 4: Strongly Polynomial Algorithm	
D	$\{i : \lvert b_i^{\mu}\rvert/\Delta \geq 1/n\}$
$E_{\mu}^{\text{Tight}}[V\backslash T]$	$\{ij \in E[V\backslash T] : \gamma_{ij}^{\mu} = 1\}$
Introduced in Chapter 5: Theorems and Proofs	
Θ^F	Set of iterations where Filtration is called
Θ^D	Set of iterations where D expands by at least one node

Table 5.2: Summary of Notations in Chapter 5

Chapter 6

Bounding the Bit Sizes

In Definition 1, I stated that all numbers computed by the algorithm must be rational numbers with bit sizes that are polynomially bounded in the bit sizes of the original inputs. This chapter fixes the algorithm presented in Chapter 4 so that f_{ij} and μ_i do not get exponentially large or small. A table of all the notations up to Chapter 6 is provided on page 94 at the end of the chapter. This will be useful for readers who are interested in a particular section rather than the full thesis.

Since our algorithm performs multiplications and divisions, it is not clear that the bit sizes of the computed numbers are polynomially bounded. The next example demonstrates how we can obtain an output that is exponential in the size of the original input after a strongly polynomial number of iterations [4]. Let $U \geq 2$ be our input.

Algorithm 7: Example of Getting Exponential Bit Size

```
1 Set counter= 1;
2 for counter = $1, \dots, n$ do
3     $U = U \times U$;
4     counter = counter +1;
5 end
6 return $U$ ;
```

The value returned by this algorithm is $U^{(2^n)}$. The bit size of the output is $2^n \log U$, and thus exponential in the bit size of the input.

In the algorithms presented in Chapters 3 and 4, the values of α, Δ, μ, and f are always rational numbers if the inputs are rational. Our goal is to round these numbers so that the numerators and denominators have reasonable bit sizes and avoid the above scenario.

6.1 Rounding our Labels

The number U has been mentioned several times throughout this thesis. Recall that in Section 2.1, we added auxiliary arcs it for each node $i \neq t$ with gain factors of $1/U$ so that $e_i = 0$ in the optimal solution. Since U is an extremely large number, the auxiliary arc is never used unless there is no other path to send the flow. In Subsection 3.4.3, we used U again to define Δ^{end}, such that $\Delta < \Delta^{\text{end}}$ implies we have optimal labels. In this section, we will also use U to help with rounding our numbers.

Let U be 2 times the product of all the numerators and denominators used to represent supplies and demands b_i and gain factors γ_{ij}. Since U is the product of at most $4m$ numbers, the bit size of U is strongly polynomial in the bit size of the largest input number. We may assume that $U \geq 500n^5$, otherwise the weakly polynomial algorithm also runs in strongly polynomial time.

By feasibility of the labels, we know that $\mu_i \leq U$ because $\gamma_{it}^{\mu} \leq 1$. However, the numerators and denominators for labels could get extremely large (e.g. $U^{(2^n)}/(U^{(2^n)}+1)$) so that their bit sizes are exponential. This also affects the bit sizes of the relabelled terms. If we could round the labels so that μ_i is always a polynomially bounded integer multiple of $1/U^4$ for all i, then the upper-bound on μ_i would help us conclude that all labels are polynomially bounded. In this section, we will round our α and our μ to achieve this effect.

We need to define two more numbers in order to do our rounding:

$$q = 40mU^4 \qquad\qquad \bar{q} = 40mU^2 = q/U^2$$

For some $a \in \mathbb{R}$, we define $\lfloor a \rfloor_q$ to be the largest integer multiple of $1/q$ such that $\lfloor a \rfloor_q \leq a$. We define $\lceil a \rceil_q$ to be the smallest integer multiple of $1/q$ such that $\lceil a \rceil_q \geq a$. These notations simply tells us whether we are rounding up or down to the next multiple of $1/q$. We can also define $\lfloor a \rfloor_{\bar{q}}$ and $\lceil a \rceil_{\bar{q}}$ similarly.

Recall that in the original Elementary Step (see Section 3.2), we computed α_1 to tighten an arc in $E[T, V\backslash T]$ and α_2 to raise some node's relabelled excess relative to the new scaling parameter Δ/α. In our updated algorithm, we will round down α_2 to a multiple of $1/q$. First compute $\alpha_{2,i}$ for each node $i \in V\backslash T - t$ according to Subsection 3.2.2. Then α_2 is the minimum of all $\alpha_{2,i}$ after rounding them to the nearest multiple of $1/q$. This could mean that $e_i'^{\mu'}$ is slightly smaller than $4(\deg(i)+2)\Delta'$ for all nodes $i \in V\backslash T$, so we will add node i to T_0 if $\alpha = \lfloor \alpha_{2,i} \rfloor_q$. Notice that we do not round α_1 because we cannot add node j to T and send flow on arc ij when ij is almost tight rather than tight. We will also round up Δ' to a multiple of $1/q$. In the proof of Claim 37, we will see that the bit sizes of α_1, α_2, and Δ are polynomial in $\log U$.

Let μ^o be the labels after the initial updates to the labels by α, and μ' be the labels after applying a new subroutine, Round-Label, to μ^o to ensure all numbers are the right sizes. Round-Label is discussed after presenting the new Elementary Step.

Algorithm 8: Elementary Step with Rounding

```
1  Compute $\alpha_1$ = minimum value to tighten an arc in $E[T, V\backslash T]$;
2  Compute $\alpha_{2,i}$ = minimum value to raise $i$'s relabelled excess to $4(\deg(i)+2)\frac{\Delta}{\alpha_2}$ for
   each $i \in V\backslash T - t$;
3  Compute $\alpha_2 = \min\{\alpha_{2,i} : i \in V\backslash T - t\}$;
4  Let $\alpha = \min\{\alpha_1, \lfloor \alpha_2 \rfloor_q\}$;
5  Let $\Delta' = \lceil \Delta/\alpha \rceil_q$;
6  for $i \in V$ do
7      if $i \in T$ then $\mu_i^o = \alpha\mu_i$;
8      if $i \in V\backslash T$ then $\mu_i^o = \mu_i$;
9  end
10 for $ij \in E$ do
11     if $ij \in E[V\backslash T, T]$ then $f'_{ij} = f_{ij}/\alpha$;
12     else if $ij \in E[V\backslash T]$ and $\gamma_{ij}^{\mu^o} < 1$ then $f'_{ij} = f_{ij}/\alpha$;
13     else $f'_{ij} = f_{ij}$;
14 end
15 for $i \in V\backslash T - t$ do
16     if $\alpha = \lfloor \alpha_{2,i} \rfloor_q$ then $T_0 = T_0 \cup \{i\}$;
17 end
18 Update $T = T \cup T_0$;
19 if $\exists i \in T_0 : e_i'^{\mu^o} < (\deg(i)+2)\Delta'$ then
20     Remove $i$ from $T_0$ ;
21     Reset $T = T_0$
22 end
23 Round-Label ($\mu^o$);
```

In the previous Elementary Step, we did not need to track the individual $\alpha_{2,i}$ to decide which nodes should be added to T_0 because nodes were added at a specific threshold. In the Elementary Step with Rounding, we need to track the $\alpha_{2,i}$ to know if nodes can be added to T_0 before reaching the threshold.

After the usual updates, we still need to round μ one more time to ensure that both the numerators and denominators of the labels are polynomially bounded. The next subroutine, Round-Label, will only increase the number of tight arcs in the network and will not affect any arcs that are already tight. The idea of Round-Label is to ensure that each μ_i is either equal to $1/\gamma(P)$ for some tight i-t path P or rounded to a multiple of $U^2/q = 1/(40mU^2)$. This can be achieved by looking at some set $S = \{t\}$ and adding nodes into S as we fix our labels. Round-Label is presented next:

Algorithm 9: Round-Label (μ^o)

```
1  Initialize $S = \{t\}$, $\mu' = \mu^o$.
2  while $S \neq V$ do
3      $\delta_1 = \min\{1/\gamma_{ij}^{\mu'} : ij \in E_{f'}^{\mu'}(\Delta'), i \in V\backslash S, j \in S\}$.
4      $\delta_2 = \min\{\frac{\lceil \mu'_i \rceil_{\bar{q}}}{\mu'_i} : i \in V\backslash S\}$.
5      $\delta = \min\{\delta_1, \delta_2\}$.
6      for $i \in V\backslash S$ do
7          $\mu'_i = \mu'_i \delta$
8      end
9      $S = S \cup \{i \in V\backslash S : \lceil \mu'_i \rceil_{\bar{q}} = \mu'_i\} \cup \{i \in V\backslash S : \exists ij \in E_{f'}^{\mu'}(\Delta'), j \in S, \text{ and } \gamma_{ij}^{\mu'} = 1\}$
10 end
```

First we need to check that Round-Label maintains a Δ'-feasible pair.

Lemma 33. *Let μ^o be the label after the original updates by Elementary Step but before Round-Label, and μ' be the labels after Round-Label. Then $\mu_i^o \leq \mu'_i \leq (1 + \frac{1}{40mU})\mu_i^o$. Furthermore, the new pair (f', μ') is a Δ'-feasible pair.*

Proof. Round-Label could increase our labels by rounding the μ^o values up or tightening arcs in $E[V\backslash S, S]$. We know that $\mu'_i \leq \lceil \mu_i^o \rceil_{\bar{q}}$ by the choice of δ_2:

$$\mu'_i \leq \mu_i^o + \frac{1}{\bar{q}} = \mu_i^o(1 + \frac{1}{\bar{q}\mu_i^o})$$

Since the labels defined in Initialization guarantees that $\mu_i^o \geq 1/U$ (see Subsection 2.2.2) and labels are monotone increasing, we can bound the above inequality by:

$$\mu'_i \leq \mu_i^o(1 + \frac{1}{\bar{q}/U}) = \mu_i^o(1 + \frac{1}{40mU})$$

Next, we want to check that (f', μ') is Δ'-feasible. Condition 1 of Definition 10 requires that f' and μ' are feasible to the primal and dual respectively. The label at t is still $\mu'_t = 1$. Since Conditions 2 requires that all $ij \in E_{f'}^{\mu'}(\Delta')$ satisfy $\gamma_{ij}^{\mu'} \leq 1$ and $E \subseteq E_{f'}^{\mu'}(\Delta')$, verifying

Condition 2 implies that μ' is dual feasible. Furthermore, Condition 3 is a stronger version of the primal feasibility and requires that $e'_i \geq R'_i$. Thus verifying Condition 3 implies that f' is primal feasible.

For Condition 2 , we know that $\delta \leq \delta_1$ ensures that tight arcs remain tight after Round-Label. For non-tight arcs, we have:

$$\frac{f'_{ij}}{\mu'_i} \leq \frac{f'_{ij}}{\mu^o_i} \leq \Delta'$$

The first inequality follows from the first part of the lemma, and the second part follows from Elementary Step returning a Δ'-feasible pair before going into Round-Label.

For Condition 3, we know that Round-Label does not modify excesses at nodes. On the other hand, we might increase the number of tight arcs, so that R'_i could decrease. Thus, $e'_i \geq R'_i$ still holds. □

6.2 Effect of Round-Label on Runtime

The bound on the increase in μ'_i due to Round-Label will help us compare $\Delta\mu_i$ before running Elementary Step with Rounding to $\Delta'\mu'_i$ after running it. Previously, $\Delta\mu_i$ decreased by a factor of α if node $i \in V\backslash T$ and stayed the same if node $i \in T$. But Round-Label increases our μ'_i slightly, so $\Delta\mu_i$ is not necessarily monotone decreasing. Instead, we need to obtain a new bound.

Lemma 34. *If we are not in the last iteration and we run the Elementary Step with Rounding, then the following must be true:*

- *For $i \in T : \Delta'\mu'_i \leq (\Delta\mu_i)(1 + \frac{1}{U})$.*
- *For $i \in V\backslash T : \Delta'\mu'_i \leq (\Delta\mu_i)\frac{1+1/U}{\alpha}$.*

Proof. We know $\Delta' = \lceil \frac{\Delta}{\alpha} \rceil_q$ and we can bound μ'_i by Lemma 33.

For $i \in T$:
During the first update in Elementary Step, we set $\mu^o_i = \alpha\mu_i$. Thus we get $\mu'_i \leq \alpha\mu_i(1 + \frac{1}{40mU})$. This gives us:

$$\begin{aligned}\Delta'\mu'_i &\leq (\lceil \frac{\Delta}{\alpha} \rceil_q)(\alpha\mu_i(1 + \frac{1}{40mU}))\\ &\leq (\frac{\Delta}{\alpha} + \frac{1}{q})(\alpha\mu_i(1 + \frac{1}{40mU}))\end{aligned}$$

Observe that we can terminate the algorithm when $\Delta' < \Delta^{\text{end}} = \frac{1}{17mU^3}$ in the strongly polynomial algorithm too if this condition precedes $|V| = 1$. Thus we have $\frac{\Delta}{\alpha} \geq \frac{1}{17mU^3} \geq \frac{2U}{q}$. This simplifies the above inequality to:

$$\begin{aligned}\Delta'\mu_i' &\leq (\frac{\Delta}{\alpha} + \frac{\Delta/\alpha}{2U})(\alpha\mu_i(1 + \frac{1}{40mU})) \\ &\leq \Delta\mu_i(1 + \frac{1}{2U})(1 + \frac{1}{40mU}) \\ &\leq \Delta\mu_i(1 + \frac{1}{U})\end{aligned}$$

For $i \in V \backslash T$:
During the first update in Elementary Step, we set $\mu_i^o = \mu_i$. Applying the same set of inequalities but with one less factor of α in the numerator gives:

$$\Delta'\mu_i' \leq (\Delta\mu_i)\frac{1 + 1/U}{\alpha}$$

□

Since $\Delta\mu_i$ is no longer monotone decreasing, using our previous definition of D (see Subsection 4.1.2) could allow nodes in D to leave. Instead, let g be the number of times where we run Elementary Step, and set $D = \{i \in V - t : \frac{|b_i^\mu|}{\Delta} \geq \frac{1}{n(1+1/U)^g}\}$.

Lemma 30 will change slightly because we could add a node to T_0 even when $e_i'^{\mu'} < 4(\deg(i) + 2)\Delta'$. This can be fixed by using $\log_{2-\epsilon}$ rather than $\log_2$ in the proof; the lemma will still hold with an increase of a small constant factor to the runtime. We need to find a new bound for Q (see Section 5.1) to determine the maximum number of iterations.

Lemma 35. *Let Q be the maximum increase to Φ in a major iteration. Then $Q = O(mn^4 \log n)$, and our algorithm has at most $O(mn^6 \log n)$ iterations.*

Proof. First let us consider how Claim 32 is affected. We could increase Γ by adding nodes to T_0 or when we run Round-Label in Elementary Step. Recall that $\Gamma_i = \log \frac{32mn\Delta}{|b_i^\mu|}$ for each $i \in D$ (see Subsection 5.1.1), where we now have $\frac{|b_i^\mu|}{\Delta} \geq \frac{1}{n(1+1/U)^g}$.

$$\begin{aligned}\Gamma_i \leq \log[(32mn)(n(1+1/U)^g)] &\leq 5 + 4\log n + g\log(1+1/U)\\ &\leq 5 + 4\log n + g/U\\ &\leq 9\log n + \frac{2nQ}{500n^5}\end{aligned}$$

The last line follows from our assumption that $U > 500n^5$. Furthermore, we know that if Q is the max increase to Φ, then there are at most $2nQ$ Elementary Steps performed by Lemma 16.

Our total increase to Γ is:

$$\begin{aligned}\Gamma &\leq \sum_{i\in V-t}(9\log n + \frac{2nQ}{500n^5})\\ &\leq 9n\log n + \frac{Q}{250n^3}\end{aligned}$$

We can apply our new bound on Γ to Lemmas 30 and 31 to see that each node i enters T_0 at most $10n\log n + \frac{Q}{250n^3}$ times. We can apply this new bound to Lemma 26:

$$\begin{aligned}Q &\leq \sum_{i\in V-t}[4(\deg(i)+2) - (\deg(i)+1)](10n\log n + \frac{Q}{250n^3})\\ &\leq (13m)(10n\log n + \frac{Q}{250n^3})\end{aligned}$$

By rearranging the above inequality, we get:

$$\begin{aligned}Q(1-\frac{13m}{250n^3}) &\leq 130mn\log n\\ Q &\leq \frac{130mn\log n}{1-\frac{13m}{250n^3}}\\ &\leq (130mn\log n)(250n^3)\end{aligned}$$

So $Q = O(mn^4\log n)$, and the number of iterations is $O(mn^6\log n)$ by replacing Q in Theorem 27. □

6.3 Bit Sizes of Numbers Computed

We want to show that all the numbers computed by the algorithm have bit sizes that are polynomial in $\log U$.

Claim 36. *All our labels are integer multiples of $U/q = 1/(40mU^3)$, and the bit sizes of our labels are polynomial in $\log U$.*

Proof. This is clearly true for sink t. Consider any other nodes and proceed by induction on the size of S. There are two reasons why node i was added to S. Firstly, node i was added because $\mu'_i = \lceil \mu'_i \rceil_{\bar{q}}$. This means that μ'_i is an integer multiple of $1/\bar{q} = U^2/q$. Otherwise, i was added because there is a tight arc ij to $j \in S$. Then there exists node $k \in S$ such that:

- Node k was added previously when $\mu'_k = \lceil \mu'_k \rceil_{\bar{q}}$ or $k = t$; and
- There is a tight path P in S from node i to node k.

Then $\mu'_i = \mu'_k/\gamma(P)$. By induction, we know that μ'_k is an integer multiple of U^2/q. We also know that U is an integer multiple of $\gamma(P)$. Thus μ'_i is an integer multiple of U/q.

Since $\mu_i \le U$, this tells us that μ_i is a rational number where the numerator is at most $40mU^4$ and its bit size is $O(\log U)$. □

It is easy to see that the bit size of Δ is always polynomially bounded in $\log U$. We know that Δ are always positive integer multiples of $1/q = 1/(40mU^4)$ from rounding. Furthermore, Initialization chose our first scaling parameter to be $\Delta^{\text{start}} = \max_{i \in V-t} e^\mu_i$ (see Subsection 3.4.2). By applying Lemma 23 with $T = \emptyset$ during Initialization, we know that $\Delta^{\text{start}} \le n \max_{j \in V-t} |b^\mu_j| \le nU$. Thus Δ is polynomially bounded.

Claim 37. *All flow values f_{ij} have bit sizes that are polynomial in $\log U$.*

Proof. Initially, $f_{ij} = 0$ for all arcs $ij \in E$. We will show that f_{ij} remains polynomially bounded after each update.

The first way that f_{ij} can be modified is through the Tight-Flow computations, which are simply maximum flow computations. Since our labels have bit sizes that are polynomial in $\log U$, the inputs to Tight-Flow (b^μ_i) are also rational numbers with bit sizes that are polynomial in $\log U$. This means that the computed f'_{ij}/μ_i, and thus f'_{ij}, must have bit sizes that are polynomially bounded after Tight-Flow.

The second way that f_{ij} can be modified is through flow augmentations. Adding (or subtracting) Δ units to f^μ_{ij} is the same as adding (or subtracting) $\Delta\mu_i$ units to f_{ij}. We know that $\Delta\mu_i$ is a polynomially bounded multiple of U/q^2 because Δ and μ_i are polynomially bounded integer multiples of $1/q$ and U/q respectively in our new Elementary Step. After a maximum of $O(mn^6 \log n)$ augmentations, the total increase (or decrease) is polynomially bounded.

Finally, we can update f_{ij} in Elementary Step by dividing f_{ij} by α. If $\alpha = \lfloor \alpha_2 \rfloor_q$, then the result clearly follows. If $\alpha = \alpha_1$, then we know that $1/\alpha = \gamma_{kl}\frac{\mu_k}{\mu_l}$ for some $kl \in E[T, V\backslash T]$ that was tightened. Now γ_{kl} is a rational number where the numerator and denominator are at most U by definition. Furthermore, Claim 36 tells us that μ_k/μ_l is a rational number where both the numerator and denominator are bounded by Uq. Thus α_1 is a rational number where both the numerator and denominator are bounded by U^2q. Since we perform Elementary Step at most $O(mn^6 \log n)$ times, f_{ij} remains polynomially bounded after this update. □

6.4 Summary

This chapter showed how we can round the labels μ and the scaling parameter Δ so that the bit sizes of the numbers that we compute are polynomially bounded in the bit sizes of the input.

The new subroutine, Round-Label, runs after every Elementary Step. This procedure would slow down our algorithm, but our algorithm still has strongly polynomial running time. Round-Label rounds all the labels so that they are polynomially bounded integer multiples of U/q, so μ_i can be written as a rational number where the denominator is between 1 and $q = 40mU^4$ for all $i \in V$. Since $\mu_i \leq U$, we know that the numerator of μ_i is an integer between 1 and $Uq = 40mU^5$. Thus the labels can be represented by rational numbers where the bit sizes are polynomially bounded by $\log U$. Next, we saw that all the f_{ij} are also polynomially bounded. This means that Végh's algorithm is a true strongly polynomial algorithm.

Notation	Meaning
Introduced in Chapter 1: Introduction	
γ_{ij}	Gain factor of arc ij, always > 0
b_i	Demands (> 0) or supplies (< 0) at node i
e_i	Excess at node i, $e_i = \sum_{ji} \gamma_{ji} f_{ji} - \sum_{ij} f_{ij} - b_i$
f_{ij}	Flow on arc ij
μ_i	Label at node i; inverse of dual solution, always > 0
E_f	The set of residual arcs: $\{ij : f_{ij} < u_{ij}\} \cup \{ji : f_{ij} > 0\}$
G_f	The residual graph with node set V and arc set E_f
Introduced in Chapter 2: Definitions and Notations	
f^{μ}_{ij}, b^{μ}_i, e^{μ}_i, γ^{μ}_{ij}	Relabelled quantities; see Subsection 2.2.2
$E^{\mu}_f(\Delta)$	The set of Δ-fat arcs: $E \cup \{ji : f^{\mu}_{ij} > \Delta\}$
$G^{\mu}_f(\Delta)$	The graph with node set V and arc set $E^{\mu}_f(\Delta)$
R_i	The reserve at i; $\sum_{ji:\gamma^{\mu}_{ji}<1} \gamma_{ji} f_{ji}$
T_0	High excess nodes; $T_0 \subseteq \{i : e^{\mu}_i \geq (\deg(i) + 2)\Delta\}$
T	Reachable nodes; $T \subseteq \{i : i \text{ is reachable from a node in } T_0 \text{ via a tight path in } E^{\mu}_f(\Delta)\}$
L	Low excess nodes; $\{i : e^{\mu}_i < (\deg(i) + 1)\Delta\}$

Notation	Meaning
Introduced in Chapter 3: Weakly Polynomial Algorithm	
α_1	Smallest scaling factor needed to tighten an arc in $E[T, V\backslash T]$
α_2	Smallest scaling factor needed to raise $e_i'^{\mu'}$ to $4(\deg(i)+2)\Delta'$ for $i \in V\backslash T$
Φ	Potential function for measuring number of iterations
Q	Total increase to Φ over all iterations
Introduced in Chapter 4: Strongly Polynomial Algorithm	
D	$\{i : \lvert b_i^\mu \rvert/\Delta \geq 1/n\}$
$E_\mu^{\text{Tight}}[V\backslash T]$	$\{ij \in E[V\backslash T] : \gamma_{ij}^\mu = 1\}$
Introduced in Chapter 5: Theorems and Proofs	
Θ^F	Set of iterations where Filtration is called
Θ^D	Set of iterations where D expands by at least one node
Introduced in Chapter 6: Bounding the Bit Sizes	
U	2 times the product of all the numerators and denominators used to represent supplies and demands b_i and gain factors γ_{ij}
q	$= 40mU^4$
$\bar{q}$	$= 40mU^2 = q/U^2$

Table 6.1: Summary of Notations in Chapter 6

Chapter 7

Conclusion

In this thesis, I presented an exposition of Végh's strongly polynomial algorithm for the generalized flow problem. This is the first strongly polynomial algorithm for the problem and resolves a long-standing open question.

In Chapter 1, we saw the standard model for generalized flow. Under the standard model, we have a capacitated network with gain factors on the arcs, and we are trying to maximize flow into a sink t. We saw the optimality conditions for the standard model as well as two earlier algorithms that solved the problem. The first algorithm is an extension of the augmenting path algorithm for traditional maximum flow problems, and thus does not run in polynomial time. The second algorithm is a scaling algorithm, which implies that it runs in weakly polynomial time. Furthermore, I showed that any linear program can be converted into an instance of the minimum cost generalized flow problem (MCGF). Being able to solve the generalized flow problem is an important step towards solving MCGF in strongly polynomial time. Finally, I gave an overview of two techniques that Végh used to obtain a strongly polynomial running time from a scaling algorithm.

In Chapter 2, we started to focus on Végh's algorithm. This required us to work on an uncapacitated network and revise the optimality conditions. I gave a more in-depth explanation of labels, which are inverses of the dual variables. Labels were used to relabel the network, and we explored the benefits of using a relabelled network over the original network. We also saw the definitions of Δ-fat arcs and Δ-feasibility, which relaxed the dual optimality conditions (i.e. conservative labels) but had stronger primal feasibility conditions (see Section 2.3). The definition of Δ-feasibility required us to maintain a pair of feasible solutions (f, μ) that became closer and closer to a pair of optimal solutions as our scaling factor Δ got smaller. Finally, we saw the definitions of high excess nodes T_0, low excess nodes L, and reachable nodes T (see Section 2.4). These nodes determine where and when we can augment flow in the algorithm.

In Chapter 3, we looked at a simpler version of Végh's algorithm which runs in weakly polynomial time. This algorithm performs one of three actions during each iteration: augment flow, expand T, or perform Elementary Step to fix the Δ-fat graph when we could not perform the previous two actions. Elementary Step helped us to either add nodes to T_0 or to identify more reachable nodes by finding tight arcs leaving T (see Section 3.2). I also gave an overview of the runtime analysis to understand why the algorithm terminates in a weakly polynomial number of iterations. Finally, we saw how to initialize a beginning flow and labels, and how to terminate with an optimal flow when we achieve optimal labels.

In Chapter 4, we looked at abundant arcs (see Section 4.1) and how we can guarantee their existence. Abundant arcs can be contracted so that we work on a smaller network. We then looked at the set D of nodes that are close to contraction (see Subsection 4.1.2), as well as the conditions required in order for Elementary Step to bring us closer to the next contraction. If Elementary Step will not bring us closer to the next contraction, then we run the new subroutine Filtration (see Section 4.2). Finally, we saw in the runtime analysis that Filtration and contracting abundant arcs meant that we could put a strongly polynomial bound on the number of iterations.

In Chapter 5, the lemmas and proofs of two major concepts were presented. First, I presented the omitted proofs from Chapter 4 that shows why Végh's algorithm terminates in a strongly polynomial number of iterations. Second, I proved that we can bound the maximum difference in flow on arcs between our current flow and the optimal (i.e. $\max_{ij\in E} |f^{\mu}_{ij} - f^{*\mu^*}_{ij}|$) flow using the relabelled excess and our scaling factor Δ.

In Chapter 6, we saw a modification of Elementary Step that ensures all numbers computed have bit sizes that are polynomially bounded in the bit sizes of the inputs. This means that Végh's algorithm is a true strongly polynomial algorithm.

Appendix A

Differences in Notation with Végh's Paper

This table summarizes all the differences in the notation and definitions between my thesis and Végh's paper [9].

Item	My Notation/Definition	Végh's Notation/Definition
Δ-fat graph	$G_f^{\mu}(\Delta)$	n/a
Low excess nodes	L	N
Close to contraction node set D	$\{i : \lvert b_i^{\mu}\rvert/\Delta \geq 1/n\}$	$\{i : \lvert b_i^{\mu}\rvert/\Delta \geq 1/16^C n\}$, $C = \#$ of contractions completed
Set of contracting iterations	n/a	$\mathcal{C}$
Set of filtrating iterations	Θ^F	$\mathcal{F}$
Set of iterations that grow D	Θ^D	$\mathcal{D}$
Scaling factor in Round-Label	δ	ϵ

Table A.1: Differences in notation with Végh's paper

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications, Chapter 15.* Prentice Hall, 1993.

[2] Andrew V. Goldberg, Serge A. Plotkin, and Éva Tardos. Combinatorial algorithms for the generalized circulation problem. *Mathematics of Operations Research*, 16(2):351–381, 1991.

[3] Dorit S. Hochbaum. Monotonizing linear programs with up to two nonzeroes per column. *Operations Research Letters*, 32(1):49–58, 2004.

[4] Thomas McCormick. Graph theory and network flows. Course Notes for OR&IE 633, Cornell University, 1999.

[5] Kenji Onaga. Dynamic programming of optimum flows in lossy communication nets. *IEEE Transactions on Circuit Theory*, 13(3):282–287, 1966.

[6] Tomasz Radzik. Approximate generalized circulation. *Technical Report 93-2, Cornell Computational Optimization Project, Cornell University*, 1993.

[7] Maiko Shigeno. A survey of combinatorial maximum flow algorithms on a network with gains. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 47(4):244–264, 2004.

[8] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of computer and system sciences*, 26(3):362–391, 1983.

[9] László A. Végh. Strongly polynomial algorithm for generalized flow maximization. *CoRR*, abs/1307.6809, 2013.