

**Definition 1 (Trust Reduction).**

Let  $A, B \in \mathcal{V}$  and  $x_i$  flow to  $N^+(A)_i$  resulting from  $\text{maxFlow}(A, B)$ ,  $u_i = \text{DTr}_{A \rightarrow N^+(A)_{i,j-1}}$ ,  $u'_i = \text{DTr}_{A \rightarrow N^+(A)_{i,j}}$ ,  $i \in [|N^+(A)|]$ ,  $j \in \mathbb{N}$ .

1. The Trust Reduction on neighbour  $i$ ,  $\delta_i$  is defined as  $\delta_i = u_i - u'_i$ .
2. The Flow Reduction on neighbour  $i$ ,  $\Delta_i$  is defined as  $\Delta_i = x_i - u'_i$ .

We will also use the standard notation for 1-norm and  $\infty$ -norm, that is:

1.  $\|\delta_i\|_1 = \sum_{i \in N^+(A)} \delta_i$
2.  $\|\delta_i\|_\infty = \max_{i \in N^+(A)} \delta_i$ .

**Definition 2 (Restricted Flow).**

Let  $A, B \in \mathcal{V}$ ,  $i \in [|N^+(A)|]$ .

1. Let  $F_{A_i \rightarrow B}$  be the flow from  $A$  to  $N^+(A)_i$  as calculated by the  $\text{maxFlow}(A, B)$  ( $x'_i$ ) when  $u'_i = u_i$ ,  $u'_k = 0 \forall k \in [|N^+(A)|] \wedge k \neq i$ .
2. Let  $S \subset N^+(A)$ . Let  $F_{A_S \rightarrow B}$  be the sum of flows from  $A$  to  $S$  as calculated by the  $\text{maxFlow}(A, B)$  ( $\sum_{i=1}^{|S|} x'_i$ ) when  $u'_C = u_C \forall C \in S$ ,  $u'_D = 0 \forall D \in N^+(A) \setminus S$ .

**Theorem 1 (Saturation theorem).**

Let  $s$  source,  $n = |N^+(s)|$ ,  $x_i, i \in [n]$ , flows to  $s$ 's neighbours as calculated by the  $\text{maxFlow}$  algorithm,  $u'_i$  new direct trusts to the  $n$  neighbours and  $x'_i$  new flows to the neighbours as calculated by the  $\text{maxFlow}$  algorithm with the new direct trusts,  $u'_i$ . It holds that  $\forall i \in [n], u'_i \leq x_i \Rightarrow x'_i = u'_i$ .

*Proof.*  $\forall i \in [n], x'_i > u'_i$  is impossible because a flow cannot be higher than its corresponding capacity. Thus  $\forall i \in [n], x'_i \leq u'_i$ . (1)

In the initial configuration of  $u_i$  and according to the flow problem setting, a combination of flows  $y_i$  such that  $\forall i \in [n], y_i = u'_i$  is a valid, albeit not necessarily maximum, configuration with a flow  $\sum_{i=1}^n y_i$ . Suppose that  $\exists k \in [n] : x'_k < u'_k$  as calculated by the  $\text{maxFlow}$  algorithm with the new direct trusts,  $u'_i$ . Then for the new  $\text{maxFlow}$   $F'$  it holds that  $F' = \sum_{i=1}^n x'_i <$

$\sum_{i=1}^n y_i$  since  $x'_k < y_k$  and (1) which is impossible because the configuration  $\forall i \in [n], x'_i = y_i$  is valid since  $\forall i \in [n], y_i = u'_i$  and also has a higher flow, thus the  $\text{maxFlow}$  algorithm will prefer the configuration with the higher flow. Thus we deduce that  $\forall i \in [n], x'_i = u'_i$ .  $\square$   $\square$

**Theorem 2 (Trust transfer theorem (flow terminology)).**

Let  $s$  source,  $t$  sink,  $n = N^+(s)$

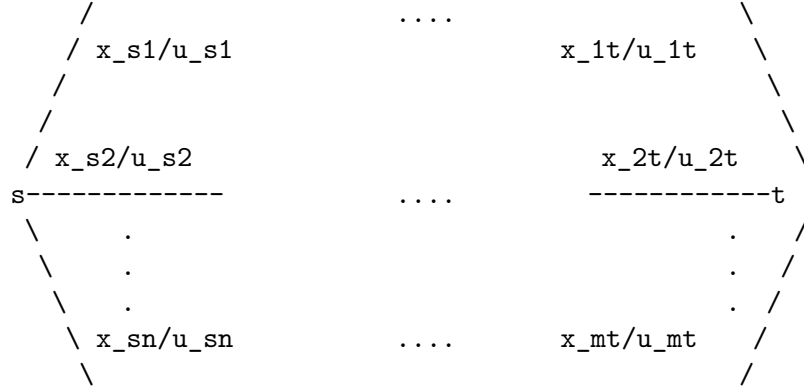
$X = \{x_1, \dots, x_n\}$  outgoing flows from  $s$ ,

$U = \{u_1, \dots, u_n\}$  outgoing capacities from  $s$ ,

$V$  the value to be transferred.

Nodes apart from  $s, t$  follow the conservative strategy.

Obviously  $\maxFlow = F = \sum_{i=1}^n x_i$ .



We create a new graph where

1.  $\sum_{i=1}^n u'_i = F - V$
2.  $\forall i \in [n] u'_i \leq x_i$

It holds that  $\maxFlow' = F' = F - V$ .

*Proof.* From theorem 1 we can see that  $x'_i = u'_i$ . It holds that  $F' = \sum_{i=1}^n x'_i = \sum_{i=1}^n u'_i = F - V$ . □ □

**Lemma 1 (Flow limit lemma).**

It is impossible for the outgoing flow  $x_i$  from  $A$  to an out neighbour of  $A$  to be greater than  $F_{A_i \rightarrow B}$ . More formally,  $x_i \leq F_{A_i \rightarrow B}$ .

*Proof.* Suppose a configuration where  $\exists i : x_i > F_{A_i \rightarrow B}$ . If we reduce the capacities  $u_k, k \neq i$  the flow that passes from  $i$  in no case has to be reduced. Thus we can set  $\forall k \neq i, u'_k = 0$  and  $u'_i = u_i$ . Then  $\forall k \neq i, x'_k = 0, x'_i = x_i$  is a valid configuration and thus by definition  $F_{A_i \rightarrow B} = x'_i = x_i > F_{A_i \rightarrow B}$ , which is a contradiction. Thus  $\forall i \in [N^+(A)], x_i \leq F_{A_i \rightarrow B}$ . □ □

**Theorem 3 (Trust-saving Theorem).**

A configuration  $U' : u'_i = F_{A_i \rightarrow B}$  for some  $i \in [|N^+(A)|]$  can yield the same  $\text{maxFlow}$  with a configuration  $U'' : u''_i = u_i, \forall k \in [|N^+(A)|], k \neq i, u''_k = u'_k$ .

*Proof.* We know that  $x_i \leq F_{A_i \rightarrow B}$  (lemma 1), thus we can see that any increase in  $u'_i$  beyond  $F_{A_i \rightarrow B}$  will not influence  $x_i$  and subsequently will not incur any change on the rest of the flows.  $\square$   $\square$

**Theorem 4 (Invariable trust reduction with naive algorithms).**

Let  $A$  source,  $n = |N^+(A)|$  and  $u'_i$  new direct trusts. If  $\forall i \in [n], u'_i \leq x_i$ , Trust Reduction  $\|\delta_i\|_1$  is independent of  $x_i, u'_i \forall$  valid configurations of  $x_i$

*Proof.* Since  $\forall i \in [n], u'_i \leq x_i$  it is (according to 1)  $x'_i = u'_i$ , thus  $\delta_i = u_i - x'_i$ . We know that  $\sum_{i=1}^n x'_i = F - V$ , so we have  $\|\delta_i\|_1 = \sum_{i=1}^n \delta_i = \sum_{i=1}^n (u_i - x'_i) = \sum_{i=1}^n u_i - F + V$  independent from  $x'_i, u'_i$   $\square$   $\square$

Here we show three naive algorithms for calculating new direct trusts so as to maintain invariable risk when paying a trusted party. To prove the correctness of the algorithms, it suffices to prove that  $\forall i \in [n] u'_i \leq x_i$  and that  $\sum_{i=1}^n u'_i = F - V$  where  $F = \sum_{i=1}^n x_i$ .

---

**Algorithm 1:** First-come, first-served trust transfer

---

**Input :**  $x_i$  flows,  $n = |N^+(s)|$ ,  $V$  value

**Output:**  $u'_i$  capacities

```

1  $F \leftarrow \sum_{i=1}^n x_i$ 
2 if  $F < V$  then
3   | return  $\perp$ 
4  $F_{cur} \leftarrow F$ 
5 for  $i \leftarrow 1$  to  $n$  do
6   |  $u'_i \leftarrow x_i$ 
7    $i \leftarrow 1$ 
8 while  $F_{cur} > F - V$  do
9   |  $reduce \leftarrow \min(x_i, F_{cur} - F + V)$ 
10  |  $F_{cur} \leftarrow F_{cur} - reduce$ 
11  |  $u'_i \leftarrow x_i - reduce$ 
12  |  $i \leftarrow i + 1$ 
13 return  $U' = \bigcup_{k=1}^n \{u'_k\}$ 

```

---

*Proof of correctness for algorithm 1.*

- We will show that  $\forall i \in [n] u'_i \leq x_i$ .  
 Let  $i \in [n]$ . In line 6 we can see that  $u'_i = x_i$  and the only other occurrence of  $u'_i$  is in line 11 where it is never increased ( $reduce \geq 0$ ), thus we see that, when returned,  $u'_i \leq x_i$ .
- We will show that  $\sum_{i=1}^n u'_i = F - V$ .  
 $F_{cur,0} = F$   
 If  $F_{cur,i} \geq F - V$ , then  $F_{cur,i+1}$  does not exist because the *while* loop breaks after calculating  $F_{cur,i}$ .  
 Else  $F_{cur,i+1} = F_{cur,i} - \min(x_{i+1}, F_{cur,i} - F + V)$ .  
 If for some  $i$ ,  $\min(x_{i+1}, F_{cur,i} - F + V) = F_{cur,i} - F + V$ , then  $F_{cur,i+1} = F - V$ , so if  $F_{cur,i+1}$  exists, then  $\forall k < i, F_{cur,k} = F_{cur,k-1} - x_k \Rightarrow$   
 $F_{cur,i} = F - \sum_{k=1}^i x_k$   
 Furthermore, if  $F_{cur,i+1} = F - V$  then  $u'_{i+1} = x_{i+1} - F_{cur,i} + F - V = x_{i+1} - F + \sum_{k=1}^{i-1} x_k + F - V = \sum_{k=1}^i x_k - V, \forall k \leq i, u'_k = 0$  and  $\forall k > i + 1, u'_k = x_k$ .  
 In total, we have  $\sum_{k=1}^n u'_k = \sum_{k=1}^i x_k - V + \sum_{k=i+1}^n x_k = \sum_{k=1}^n x_k - V \Rightarrow$   
 $\sum_{k=1}^n u'_k = F - V$ .

□

□

*Complexity of algorithm 1.*

First we will prove that on line 13  $i \leq n + 1$ . Suppose that  $i > n + 1$  on line 13. This means that  $F_{cur,n}$  exists and  $F_{cur,n} = F - \sum_{i=1}^n x_i = 0 \leq F - V$  since, according to the condition on line 2,  $F - V \geq 0$ . This means however that the *while* loop on line 8 will break, thus  $F_{cur,n+1}$  cannot exist and  $i = n + 1$  on line 13, which is a contradiction, thus  $i \leq n + 1$  on line 13. Since  $i$  is incremented by 1 on every iteration of the *while* loop (line 12), the complexity of the *while* loop is  $O(n)$  in the worst case. The complexity of lines 2 - 4 and 7 is  $O(1)$  and the complexity of lines 1, 5 - 6 and 13 is  $O(n)$ , thus the total complexity of algorithm 1 is  $O(n)$ . □ □

---

**Algorithm 2:** Absolute equality trust transfer ( $\|\Delta_i\|_\infty$  minimizer)

---

**Input** :  $x_i$  flows,  $n = |N^+(s)|$ ,  $V$  value  
**Output:**  $u'_i$  capacities

```

1  $F \leftarrow \sum_{i=1}^n x_i$ 
2 if  $F < V$  then
3   | return  $\perp$ 
4 for  $i \leftarrow 1$  to  $n$  do
5   |  $u'_i \leftarrow x_i$ 
6  $reduce \leftarrow \frac{V}{n}$ 
7  $reduction \leftarrow 0$ 
8  $empty \leftarrow 0$ 
9  $i \leftarrow 0$ 
10 while  $reduction < V$  do
11   | if  $u'_i > 0$  then
12     | if  $x_i < reduce$  then
13       |  $empty \leftarrow empty + 1$ 
14       | if  $empty < n$  then
15         |  $reduce \leftarrow reduce + \frac{reduce - x_i}{n - empty}$ 
16       |  $reduction \leftarrow reduction + u'_i$ 
17       |  $u'_i \leftarrow 0$ 
18     | else if  $x_i \geq reduce$  then
19       |  $reduction \leftarrow reduction + u'_i - (x_i - reduce)$ 
20       |  $u'_i \leftarrow x_i - reduce$ 
21   |  $i \leftarrow (i + 1) \bmod n$ 
22 return  $U' = \bigcup_{k=1}^n \{u'_k\}$ 

```

---

We will start by showing some results useful for the following proofs. Let  $j$  be the number of iterations of the **while** loop for the rest of the proofs for algorithm 2 (think of  $i$  from line 21 without the  $\bmod n$ ).

First we will show that  $empty \leq n$ .  $empty$  is only modified on line 13 where it is incremented by 1. This happens only when  $u'_i > 0$  (line 11), which is assigned the value 0 on line 17. We can see that the incrementation of  $empty$  can happen at most  $n$  times because  $|U'| = n$ . Since  $empty_0 = 0$ ,  $empty \leq n$  at all times of the execution.

Next we will derive the recursive formulas for the various variables.

$empty_0 = 0$

$$\begin{aligned}
empty_{j+1} &= \begin{cases} empty_j, & u'_{(j+1) \bmod n} = 0 \\ empty_j + 1, & u'_{(j+1) \bmod n} > 0 \wedge x_{(j+1) \bmod n} < reduce_j \\ empty_j, & u'_{(j+1) \bmod n} > 0 \wedge x_{(j+1) \bmod n} \geq reduce_j \end{cases} \\
reduce_0 &= \frac{V}{n} \\
reduce_{j+1} &= \begin{cases} reduce_j, & u'_{(j+1) \bmod n} = 0 \\ reduce_j + \frac{reduce_j - x_{(j+1) \bmod n}}{n - empty_{j+1}}, & u'_{(j+1) \bmod n} > 0 \wedge x_{(j+1) \bmod n} < reduce_j \\ reduce_j, & u'_{(j+1) \bmod n} > 0 \wedge x_{(j+1) \bmod n} \geq reduce_j \end{cases} \\
reduction_0 &= 0 \\
reduction_{j+1} &= \begin{cases} reduction_j, & u'_{(j+1) \bmod n} = 0 \\ reduction_j + u'_{(j+1) \bmod n}, & u'_{(j+1) \bmod n} > 0 \wedge x_{(j+1) \bmod n} < reduce_j \\ reduction_j + u'_{(j+1) \bmod n} - x_{(j+1) \bmod n} + reduce_{j+1}, & u'_{(j+1) \bmod n} > 0 \wedge x_{(j+1) \bmod n} \geq reduce_j \end{cases}
\end{aligned}$$

In the end,  $r = reduce$  is such that  $r = \frac{\sum_{x \in S} x}{n - |S|}$  where  $S = \{\text{flows } y \text{ from } s \text{ to } N^+(s) \text{ according to } mas$   
 $y < r\}$ . Also,  $\sum_{i=1}^n u'_i = \sum_{i=1}^n \max(0, x_i - r)$ . TOPROVE

*Proof of correctness for algorithm 2.*

- We will show that  $\forall i \in [n] u'_i \leq x_i$ .  
On line 5,  $\forall i \in [n] u'_i = x_i$ . Subsequently  $u'_i$  is modified on line 17, where it becomes equal to 0 and on line 20, where it is assigned  $x_i - reduce$ . It holds that  $x_i - reduce \leq x_i$  because initially  $reduce = \frac{V}{n} \geq 0$  and subsequently  $reduce$  is modified only on line 15 where it is increased ( $n > empty$  because of line 14 and  $reduce > x_i$  because of line 12, thus  $\frac{reduce - x_i}{n - empty} > 0$ ). We see that  $\forall i \in [n], u'_i \leq x_i$ .
- We will show that  $\sum_{i=1}^n u'_i = F - V$ .  
The variable  $reduction$  keeps track of the total reduction that has happened and breaks the **while** loop when  $reduction \geq V$ . We will first show that  $reduction = \sum_{i=1}^n (x_i - u'_i)$  at all times and then we will prove that  $reduction = V$  at the end of the execution. Thus we will have proven that  $\sum_{i=1}^n u'_i = \sum_{i=1}^n x_i - V = F - V$ .
  - On line 5,  $u'_i = x_i \Rightarrow \sum_{i=1}^n (x_i - u'_i) = 0$  and  $reduction = 0$ .
On line 17,  $u'_i$  is reduced to 0 thus  $\sum_{i=1}^n (x_i - u'_i)$  is increased by  $u'_i$ . Similarly, on line 16  $reduction$  is increased by  $u'_i$ , the same as the

increase in  $\sum_{i=1}^n (x_i - u'_i)$ .

On line 20,  $u'_i$  is reduced by  $u'_i - x_i + \text{reduce}$  thus  $\sum_{i=1}^n (x_i - u'_i)$  is increased by  $u'_i - x_i + \text{reduce}$ . On line 19,  $\text{reduction}$  is increased by  $u'_i - x_i + \text{reduce}$ , which is equal to the increase in  $\sum_{i=1}^n (x_i - u'_i)$ .

We also have to note that neither  $u'_i$  nor  $\text{reduction}$  is modified in any other way from line 10 and on, thus we conclude that  $\text{reduction} = \sum_{i=1}^n (x_i - u'_i)$  at all times.

- Suppose that  $\text{reduction}_j > V$  on the line 22. Since  $\text{reduction}_j$  exists,  $\text{reduction}_{j-1} < V$ . If  $x_{j \bmod n} < \text{reduce}_{j-1}$  then  $\text{reduction}_j = \text{reduction}_{j-1} + u'_{j \bmod n}$ . Since  $\text{reduction}_j > V$ ,  $u'_{j \bmod n} > V - \text{reduction}_{j-1}$ . TOCOMPLETE

□

□

#### *Complexity of algorithm 2.*

In the worst case scenario, each time we iterate over all capacities only the last non-zero capacity will become zero and every non-zero capacity must be recalculated. This means that every  $n$  steps exactly 1 capacity becomes zero and eventually all capacities (maybe except for one) become zero. Thus we need  $O(n^2)$  steps in the worst case. □ □

A variation of this algorithm using a Fibonacci heap with complexity  $O(n)$  can be created, but that is part of further research.

#### *Proof that algorithm 2 minimizes the $\|\Delta_i\|_\infty$ norm.*

Suppose that  $U'$  is the result of an execution of algorithm 2 that does not minimize the  $\|\Delta_i\|_\infty$  norm. Suppose that  $W$  is a valid solution that minimizes the  $\|\Delta_i\|_\infty$  norm. Let  $\delta$  be the minimum value of this norm. There exists  $i \in [n]$  such that  $x_i - w_i = \delta$  and  $u'_i < w_i$ . Because both  $U'$  and  $W$  are valid solutions ( $\sum_{i=1}^n u'_i = \sum_{i=1}^n w_i = F - V$ ), there must exist a set  $S \subset U'$  such that  $\forall u'_j \in S, u'_j > w_j$  TOCOMPLETE. □

---

**Algorithm 3:** Proportional equality trust transfer

---

**Input** :  $x_i$  flows,  $n = |N^+(s)|$ ,  $V$  value

**Output:**  $u'_i$  capacities

```

1  $F \leftarrow \sum_{i=1}^n x_i$ 
2 if  $F < V$  then
3   | return  $\perp$ 
4 for  $i \leftarrow 1$  to  $n$  do
5   |  $u'_i \leftarrow x_i - \frac{V}{F}x_i$ 
6 return  $U' = \bigcup_{k=1}^n \{u'_k\}$ 

```

---

*Proof of correctness for algorithm 3.*

– We will show that  $\forall i \in [n] \ u'_i \leq x_i$ .

According to line 5, which is the only line where  $u'_i$  is changed,  $u'_i = x_i - \frac{V}{F}x_i \leq x_i$  since  $x_i, V, F > 0$  and  $V \leq F$ .

– We will show that  $\sum_{i=1}^n u'_i = F - V$ .

With  $F = \sum_{i=1}^n x_i$ , on line 6 it holds that  $\sum_{i=1}^n u'_i = \sum_{i=1}^n (x_i - \frac{V}{F}x_i) = \sum_{i=1}^n x_i - \frac{V}{F} \sum_{i=1}^n x_i = F - V$ .

□

□

*Complexity of algorithm 3.*

The complexity of lines 1, 4 - 5 and 6 is  $O(n)$  and the complexity of lines 2 - 3 is  $O(1)$ , thus the total complexity of algorithm 3 is  $O(n)$ . □ □

Naive algorithms result in  $u'_i \leq x_i$ , thus according to 4,  $\|\delta_i\|_1$  is invariable for any of the possible solutions  $U'$ , which is not necessarily the minimum (usually it will be the maximum). The following algorithms



concentrate on minimizing two  $\delta_i$  norms,  $\|\delta_i\|_\infty$  and  $\|\delta_i\|_1$ .

---

**Algorithm 4:**  $\|\delta_i\|_\infty$  minimizer

---

**Input** :  $X = \{x_i\}$  flows,  $n = |N^+(s)|$ ,  $V$  value,  $\epsilon_1, \epsilon_2$   
**Output:**  $u'_i$  capacities

```

1 if  $\epsilon_1 < 0 \vee \epsilon_2 < 0$  then
2   | return  $\perp$ 
3  $F \leftarrow \sum_{i=1}^n x_i$ 
4 if  $F < V$  then
5   | return  $\perp$ 
6  $\delta_{max} \leftarrow \max_{i \in [n]} \{u_i\}$ 
7  $\delta^* \leftarrow \text{BinSearch}(0, \delta_{max}, F - V, n, X, \epsilon_1, \epsilon_2)$ 
8 for  $i \leftarrow 1$  to  $n$  do
9   |  $u'_i \leftarrow \max(u_i - \delta^*, 0)$ 
10 return  $U' = \bigcup_{k=1}^n \{u'_k\}$ 

```

---

Since trust should be considered as a continuous unit and binary search dissects the possible interval for the solution on each recursive call, inclusion of the  $\epsilon$ -parameters in **BinSearch** is necessary for the algorithm to complete in a finite number of steps.

---

**Algorithm 5:** \*

---

**Input** :  $bot, top, F', n, X, \epsilon_1, \epsilon_2$   
**Output:**  $\delta^*$

```

1 function BinSearch if  $bot = top$  then
2   | return  $bot$ 
3 else
4   | for  $i \leftarrow 1$  to  $n$  do
5     |  $u'_i \leftarrow \max(0, u_i - \frac{top+bot}{2})$ 
6   | if  $maxFlow < F' - \epsilon_1$  then
7     | return BinSearch( $bot, \frac{top+bot}{2}, F', n, X, \epsilon_1, \epsilon_2$ )
8   | else if  $maxFlow > F' + \epsilon_2$  then
9     | return BinSearch( $\frac{top+bot}{2}, top, F', n, X, \epsilon_1, \epsilon_2$ )
10  | else
11    | return  $\frac{top+bot}{2}$ 

```

---

*Proof that  $maxFlow(\delta)$  is strictly decreasing for  $\delta : maxflow(\delta) < F$ .*  
Let  $maxFlow(\delta)$  be the  $maxFlow$  with  $\forall i \in [n], u'_i = \max(0, u_i - \delta)$ .  
We will prove that the function  $maxFlow(\delta)$  is strictly decreasing for all

$\delta \leq \max_{i \in [n]} \{u_i\}$  such that  $\maxFlow(\delta) < F$ .

Suppose that  $\exists \delta_1, \delta_2 : \delta_1 < \delta_2 \wedge \maxFlow(\delta_1) \leq \maxFlow(\delta_2) < F$ . We will work with configurations of  $x'_{i,j}$  such that  $x'_{i,j} \leq x_i, j \in \{1, 2\}$ .

Let  $S_j = \{i \in N^+(s) : i \in MinCut_j\}$ . It holds that  $S_1 \neq \emptyset$  because otherwise  $MinCut_1 = MinCut_{\delta=0}$  which is a contradiction because then  $\maxFlow(\delta_1) = F$ . Moreover, it holds that  $S_1 \subseteq S_2$ , since  $\forall u'_{i,2} > 0, u'_{i,2} < u'_{i,1}$ . Every node in the  $MinCut_j$  is saturated, thus  $\forall i \in S_1, x'_{i,j} = u'_{i,j}$ . Thus  $\sum_{i \in S_1} x_{i,2} < \sum_{i \in S_1} x_{i,1}$  and, since  $\maxFlow(\delta_1) \leq \maxFlow(\delta_2)$ ,

we conclude that for the same configurations,  $\sum_{i \in N^+(s) \setminus S_1} x_{i,2} > \sum_{i \in N^+(s) \setminus S_1} x_{i,1}$ .

However, since  $x'_{i,j} \leq x_i, j \in \{1, 2\}$ , the configuration  $[x''_{i,1} = x'_{i,2}, i \in N^+(s) \setminus S_1], [x''_{i,1} = x'_{i,1}, i \in S_1]$  is valid for  $\delta = \delta_1$  and then  $\sum_{i \in S_1} x''_{i,1} +$

$\sum_{i \in N^+(s) \setminus S_1} x''_{i,1} = \sum_{i \in S_1} x'_{i,1} + \sum_{i \in N^+(s) \setminus S_1} x'_{i,2} > \maxFlow(\delta_1)$ , contradiction.

Thus  $\maxFlow(\delta)$  is strictly decreasing.  $\square$   $\square$

We can see that if  $V > 0, F' = F - V < F$  thus if  $\delta \in (0, \max_{i \in [n]} \{u_i\}] :$   
 $\maxFlow(\delta) = F' \Rightarrow \delta = \min \|\delta_i\|_\infty : \maxFlow(\|\delta_i\|_\infty) = F'$ .

*Proof of correctness for function 5.*

Supposing that  $[F' - \epsilon_1, F' + \epsilon_2] \subset [\maxFlow(top), \maxFlow(bot)]$ , or equivalently  $\maxFlow(top) \leq F' - \epsilon_1 \wedge \maxFlow(bot) \geq F' + \epsilon_2$ , we will prove that  $\maxFlow(\delta^*) \in [F' - \epsilon_1, F' + \epsilon_2]$ .

First of all, we should note that if an invocation of **BinSearch** returns without calling **BinSearch** again (line 2 or 11), its return value will be equal to the return value of the initial invocation of **BinSearch**, as we can see on lines 7 and 9, where the return value of the invoked **BinSearch** is returned without any modification. The case where **BinSearch** is called again is analyzed next:

- If  $\maxFlow(\frac{top+bot}{2}) < F' - \epsilon_1 < F'$  (line 6) then, since  $\maxFlow(\delta)$  is strictly decreasing,  $\delta^* \in [bot, \frac{top+bot}{2})$ . As we see on line 7, the interval  $(\frac{top+bot}{2}, top]$  is discarded when the next **BinSearch** is called. Since  $F' + \epsilon_2 \leq \maxFlow(bot)$ , we have  $[F' - \epsilon_1, F' + \epsilon_2] \subset [\maxFlow(\frac{top+bot}{2}), \maxFlow(bot)]$  and the length of the available interval is divided by 2.
- Similarly, if  $\maxFlow(\frac{top+bot}{2}) > F' + \epsilon_2 > F'$  (line 8) then  $\delta^* \in (\frac{top+bot}{2}, top]$ . According to line 9, the interval  $[bot, \frac{top+bot}{2})$  is discarded when the next **BinSearch** is called. Since  $F' - \epsilon_1 \geq \maxFlow(top)$ , we have  $[F' - \epsilon_1, F' + \epsilon_2] \subset (\maxFlow(top), \maxFlow(\frac{top+bot}{2})]$  and the length of the available interval is divided by 2.

As we saw,  $[F' - \epsilon_1, F' + \epsilon_2] \subset [\maxFlow(top), \maxFlow(bot)]$  in every recursive call and  $top - bot$  is divided by 2 in every call. From topology we know that  $A \subset B \Rightarrow |A| < |B|$ , so the recursive calls cannot continue infinitely.  $|[F' - \epsilon_1, F' + \epsilon_2]| = \epsilon_1 + \epsilon_2$ . Let  $bot_0, top_0$  the input values given to the initial invocation of **BinSearch**,  $bot_j, top_j$  the input values given to the  $j$ -th recursive call of **BinSearch** and  $len_j = |[bot_j, top_j]| = top_j - bot_j$ . We have  $\forall j > 0, len_j = top_j - bot_j = \frac{top_{j-1} - bot_{j-1}}{2} \Rightarrow \forall j > 0, len_j = \frac{top_0 - bot_0}{2^j}$ . We understand that in the worst case  $len_j = \epsilon_1 + \epsilon_2 \Rightarrow 2^j = \frac{top_0 - bot_0}{\epsilon_1 + \epsilon_2} \Rightarrow j = \log_2(\frac{top_0 - bot_0}{\epsilon_1 + \epsilon_2})$ . Also, as we saw earlier,  $\delta^*$  is always in the available interval, thus  $\maxFlow(\delta^*) \in [F' - \epsilon_1, F' + \epsilon_2]$ .  $\square$   $\square$

*Complexity of function 5.*

Lines 1 - 2 have complexity  $O(1)$ , lines 4 - 5 have complexity  $O(n)$ , lines 6 - 11 have complexity  $O(\maxFlow) + O(\text{BinSearch})$ . As we saw in the proof of correctness for function 5, we need at most  $\log_2(\frac{top - bot}{\epsilon_1 + \epsilon_2})$  recursive calls of **BinSearch**. Thus the function 5 has worst-case complexity  $O((\maxFlow + n) \log_2(\frac{top - bot}{\epsilon_1 + \epsilon_2}))$ .  $\square$   $\square$

*Proof of correctness for algorithm 4.*

We will show that  $\maxFlow \in [F - V - \epsilon_1, F - V + \epsilon_2]$ , with  $u'_i$  decided by algorithm 4.

Obviously  $\maxFlow(0) = F, \maxFlow(\max_{i \in [n]} \{u_i\}) = 0$ , thus  $\delta^* \in \max_{i \in [n]} \{u_i\}$ .

According to the proof of correctness for function 5, we can directly see that  $\maxFlow(\delta^*) \in [F - V - \epsilon_1, F - V + \epsilon_2]$ , given that  $\epsilon_1, \epsilon_2$  are chosen so that  $F - V - \epsilon_1 \geq 0, F - V + \epsilon_2 \leq F$ , so as to satisfy the condition  $[F' - \epsilon_1, F' + \epsilon_2] \subset [\maxFlow(top), \maxFlow(bot)]$ .  $\square$   $\square$

*Complexity of algorithm 4.*

The complexity of lines 1 - 2 and 4 - 5 is  $O(1)$ , the complexity of lines 3, 6, 8 - 9 and 10 is  $O(n)$  and the complexity of line 7 is  $O(\text{BinSearch}) = O((\maxFlow + n) \log_2(\frac{\delta_{max}}{\epsilon_1 + \epsilon_2}))$ , thus the total complexity of algorithm 4 is  $O((\maxFlow + n) \log_2(\frac{\delta_{max}}{\epsilon_1 + \epsilon_2}))$ .  $\square$   $\square$

However, we need to minimize  $\sum_{i=1}^n (u_i - u'_i) = \|\delta_i\|_1$ .