PUSH-RELABEL MAX-FLOW ALGORITHM

Presented by Kenzie MacNeil

Submitted to Professor Anil Maheshwari
for the course COMP 5703 - Advanced Algorithms
Fall 2014
DISCLAIMER

The information contained within this slide deck presentation has been derived from the works listed in the bibliography.

This slide deck is not to be considered original research. It is simply the presentation of reconstituted research, thoughts and ideas from other authors.
PRESENTATION OVERVIEW

- Introduction
- Definitions
- Algorithm
- Example
- Termination and Correctness
- Conclusion
- Questions and Comments
The Maximum Flow problem is the attempt to find the flow of maximum value in a given flow network.

Flow Networks consist of:
- A finite directed graph $G = (V, E)$,
- A vertex called the source, $s$,
- A vertex called the sink, $t$, and
- A non-negative capacity, $c(e)$, for each edge.

Paraphrased from [1].
A flow function is a real-valued function which satisfies three properties:

- **Capacity Constraint:**
  For all $u, v \in V$, $f(u, v) \leq c(u, v)$

- **Skew Symmetry:**
  For all $u, v \in V$, $f(u, v) = -f(v, u)$

- **Flow Conservation:**
  For all $u \in V - \{s, t\}$, $\sum_{u \in V} f(u, v) = 0$
Residual Network consists of edges which can admit more flow given a residual capacity, 
$c_f(u, v) = c(u, v) - f(u, v)$.

Augmenting path is a path from $s$ to $t$ along which additional flow can be sent.
GENERIC PUSH-RELABEL MAXIMUM FLOW ALGORITHM
Push-Relabel Algorithm

History

- Push-Relabel algorithm was originally developed by Andrew V. Goldberg and Robert E. Tarjan.
- Algorithm and results were published in the mid-to-late 1980s, [2, 3].

- Previous maximum flow algorithms, such as Ford-Fulkerson, used the concept of residual network graphs and augmenting paths to determine max flow.

- Push-Relabel used a concept of a preflow to determine the max flow instead of an augmenting path.
- Preflow is a concept originally developed by A. V. Karzanov, [4].

- Sometimes referred to as the Preflow-Push algorithm.
**Push-Relabel Algorithm**

**Introduction**

- The Push-Relabel algorithm:
  - Works at converting a preflow, \( f \), to a normal flow.
  - Terminates once the preflow becomes a flow.

- This final flow produced at termination also turns out to be the maximum flow.

- Goldberg and Tarjan defined a generic Push-Relabel algorithm which solves the max flow problem.
  - No implementation details.
  - Other versions of the algorithm improve upon the time bound of this generic algorithm.
Preflow is a real-valued function $f$ on vertex pairs. Total flow into a vertex can exceed the flow out of the vertex. A preflow where all $v \in V - \{s, t\}$ have a flow excess of zero, $e_f(v) = 0$, is a normal flow.

![Flow Excess = 3](image)

Fig: Simple example of a preflow at vertex $v$.

The preflow function is also referred to as $s-t$ preflow.
Preflows must satisfy:

- **Capacity Constraint:**
  For all $u, v \in V$, $f(u,v) \leq c(u, v)$

- **Skew Symmetry:**
  For all $u, v \in V$, $f(u, v) = -f(v, u)$

- **Flow Conservation:**
  For all $u \in V - \{s, t\}$, $\sum_{u \in V} f(u, v) = 0$
**Push-Relabel Algorithm Preflow**

Preflows must satisfy:

- **Capacity Constraint:**
  
  For all $u, v \in V$, $f(u, v) \leq c(u, v)$

- **Skew Symmetry:**
  
  For all $u, v \in V$, $f(u, v) = -f(v, u)$

- **Non-Negative Constraint:**
  
  The flow into $v \in V - \{s\}$ must be greater or equal to the total flow out of $v$.

  For all $u \in V, v \in V - \{s\}$, $\sum f(u, v) \geq 0$

From [3, 5].
**Push-Relabel Algorithm**

**Preflow**

- Flow Excess, $e_f(v)$, is the net flow into $v$ where $v \in V$ for some preflow $f$,

$$
e_f(x) = \begin{cases} \sum_{u \in V} f(u, v), & v \in V - \{s\} \\ \infty, & v = s \end{cases}
$$

- Active Vertex is a vertex $v$ which satisfies all of the properties:
  - Is not the source or sink, $v \in V - \{s,t\}$;
  - Positive flow excess, $e_f(v) > 0$; and
  - Has a valid label value, $d(v) < \infty$.

From [3, 5].

Presented by Kenzie MacNeil
Submitted to Prof. Anil Maheshwari for COMP 5703
Push-Relabel Algorithm

Preflow

- Push-Relabel algorithm also uses the concepts of a residual graph.

- Residual Graph for a preflow $f$ is defined as $G_f = (V, E_f)$.

- Residual Capacity for a preflow $f$ is defined as $r_f(v, w) = c(v, w) - f(v, w)$.

- Residual Edges for a preflow $f$ is defined as the set of edges with positive residual capacity, $E_f = \{ (v, w) \mid r_f(v, w) > 0 \}$.

From [3].

Presented by Kenzie MacNeil
Submitted to Prof. Anil Maheshwari for COMP 5703
**Generic Push-Relabel Algorithm Labeling**

- Push-Relabel also uses a *valid labeling* function, $d$, to determine which vertex pair should be selected for the push operation.

- A valid labeling $d$ is a nonnegative integer function applied to all vertices to denote a label.

- The labeling is often referred to as a height or distance.

- This function is sometimes compared to the physical intuition that liquids naturally flow downhill.

From [3, 5].
Generic Push-Relabel Algorithm Labeling

Valid labeling for a preflow consists of:
- For \( v \in V \), \( 0 \leq d(v) \leq \infty \);
- \( d(s) = |V| \); (source condition)
- \( d(t) = 0 \); (sink condition)
- \( d(v) \leq d(w) + 1 \) for every residual edge \((v,w) \in E_f\).

A labeling \( d \) and a preflow \( f \) are said to be compatible if \( d \) adheres to the properties above.
PUSH-RELABEL ALGORITHM
OVERVIEW

- The algorithm pushes local flow excess starting at the source, \( s \), along the vertices towards the sink, \( t \).

- The algorithm maintains a compatible vertex labeling function, \( d \), to the preflow \( f \).

- The labeling is used to determine where to push the flow excess.

- The algorithm repeatedly performs either a push operation or a relabel operation so long as there is an active vertex in \( G_f \).
**Generic Push-Relabel Algorithm**

**Push Operation**

- The push operation is used to move flow excess from one vertex to another.

- The transfer of excess can be performed across a vertex pair \((v, w) \in E_f\) if:
  - \(v\) is an active vertex;
  - The edge has a positive residual capacity, \(r_f(v, w) > 0\);
  - Label distance is \(d(v) = d(w) + 1\)

- This allows the algorithm to move a delta excess flow, \(\delta = \min(\epsilon_f(v), r_f(v, w))\), from \(v\) to \(w\).
**Generic Push-Relabel Algorithm**

**Push Operation**

- A push from v to w is considered saturating if no more flow can be sent over the edge:
  \[ \delta = r_f(v, w) \]

- A push from v to w is non-saturating if all the excess from v the push over the edge and the edge still has some residual capacity:
  \[ \delta = e_f(v) \]

From [3].
Generic Push-Relabel Algorithm

**Push Operation**

**Input:** Preflow $f$, labels $d$ and $(v, w)$ where $v, w \in V$

**Output:** Preflow $f$

**Applicable** if $v \in V - \{s, t\}$, $d(v) < \infty$, $e_f(v) > 0$, $r_f(v, w) > 0$ and $d(v) = d(w) + 1$

Begin

\[ \delta := \min( e_f(v), r_f(v, w) ) \]

\[ f(v, w) := f(v, w) + \delta \]

\[ f(w, v) := f(w, v) - \delta \]

\[ e_f(v) := e_f(v) - \delta \]

\[ e_f(w) := e_f(w) + \delta \]

Return $f$;

End

Pseudocode adapted [3].
**Generic Push-Relabel Algorithm Relabeling Operation**

- The relabel operation is used to increase the label value of a single active vertex so that excess flow can be pushed out of the active vertex.

- The relabel operation is performed when all residual edges out of the active vertex have a positive residual capacity, $r_f(v, w) > 0$.
  - This implies that $v$’s label is smaller or equal to all vertices $w$, $d(v) \leq d(w)$.
  - Meaning no push operation across these edges was possible given the push condition $d(v) = d(w) + 1$.  

From [3].

Presented by Kenzie MacNeil
Submitted to Prof. Anil Maheshwari for COMP 5703

Fall 2014
**Generic Push-Relabel Algorithm Relabeling Operation**

- The relabel operation for some active vertex $v$ selects the smallest label from the vertices with positive residual edges, $r_f(v, w) > 0$.

- The active vertex is then assigned this smallest label value + 1 such that $d(v) := \min\{d(w) + 1 \mid (v, w) \in E_f\}$.

- This will allow the active vertex $v$ to potentially push its excess flow to at least one other vertex during the algorithm’s next iteration.

From [3].
**Generic Push-Relabel Algorithm**

**Relabeling Operation**

**Input:** Preflow $f$, labels $d$ and $v \in V - \{s, t\}$

**Output:** Labels $d$

**Applicable if** $v \in V - \{s, t\}$, $d(v) < \infty$, $e_f(v) > 0$ and $\forall w \in V$, $r_f(v, w) > 0$ which implies $d(v) \leq d(w)$

**Begin**

```
if \(\{(v,w) \in E_f\} \neq \emptyset\) then
    d(v) := \min( d(w) + 1 | (v, w) \in E_f )
else
    d(v) := \infty
endif
```

return $d$;

**End**

Pseudocode adapted [3].
**Generic Push-Relabel Algorithm Initialization**

- The algorithm initializes the following values in the residual graph before performing the push and relabel operations in the main loop:
  - Initializes the preflow of all edges in the residual graph.

- Initialize the labeling such that:
  - \( d(s) = |V| \); and
  - \( d(v) = 0 \) for \( v \in V - \{s\} \).

- Performs saturating pushes along all residual edges out of the source, \( (s, v) \in E_f \) and \( v \in V \).
Once complete, the algorithm repeatedly performs either the push or relabel operations against active vertices.

The algorithm continues until no applicable operation can performed.
- The algorithm terminates when there are no more active vertices.

From [3].
**Generic Push-Relabel Algorithm Pseudocode**

**Input:** Network Flow Graph $G = (V, E)$, $s$, $t$ and $c$

**Output:** A maximum flow $f$

Begin

forall the $(v,w) \in (V - \{s\}) (V - \{s\})$ do

$f(v,w) := 0$

$f(w,v) := 0$

end

forall the $v \in V$ do

$f(s,v) := r_f(s,v)$

$f(v,s) := -r_f(s,v)$

end

$d(s) := |V|$

forall the $v \in V - \{s\}$ do

$d(v) := 0$

$e_f(v) := f(s,v)$

end

* The pseudocode continues onto the next slide.

Pseudocode adapted [3].
Generic Push-Relabel Algorithm Pseudocode (cont’d)

/* Loop while there exists an active vertex */
while ∃ v ∈ V – {s, t} with either applicable PUSH or RELABEL operation do

    Perform either a PUSH() or a RELABEL() on v.

End

return f;

End

- Lets apply this algorithm to the network flow graph from [6] found in the course notes [1].
- Note that we have yet to define which active vertex is to be selected for the Push or Relabel operations.

Pseudocode adapted [3].
Push-Relabel Example

Initial Network Flow Graph

\[ G = \]

Network flow graph from [6].
PUSH-RELABEL EXAMPLE
INITIALIZATION

$G_f =$

Initialize labels and push initial excess out of the source.
Push-Relabel Example
Active Vertex A

\[ G_f = \]

Relabel \( a \) then push excess along \((a, b), (a, d) \in E_f\).
Relabel \( a \) then push excess back to the source.
Push-Relabel Example
Active Vertex B

$G_f =$

Relabel $b$ then excess along $(b, t), (b, c) \in E_f.$
Presented by Kenzie MacNeil
Submitted to Prof. Anil Maheshwari for COMP 5703

**Push-Relabel Example**

**Active Vertex C**

\[ G_f = \]

Relabel c then excess along \((c, d) \in E_f\).
**Push-Relabel Example**

**Active Vertex D**

\[ G_f = \]

\[ a \]
\[ d = 7 \]
\[ e = 0 \]

\[ b \]
\[ d = 1 \]
\[ e = 2 \]

\[ s \]
\[ d = 6 \]

\[ c \]
\[ d = 1 \]
\[ e = 0 \]

\[ t \]
\[ d = 0 \]

Relabel \( d \) then excess along \((d, t) \in E_f\).
**Push-Relabel Example**

**Active Vertex B**

\[ G_f = \]

Relabel \( b \) then excess along \((b, a) \in E_f\).
Push the remaining excess from $a$ back to the source $s$. 
Generic Push-Relabel Algorithm Analysis
Correctness and Termination
**Generic Push-Relabel Analysis**

- The following theorems, lemmas and proofs have been taken from Goldberg-Tarjan’s “A New Approach to the Maximum-Flow Problem”, [3], and Kleinberg-Tardos’ “Algorithm Design”, [5].

- With them we can prove that the generic Push-Relabel algorithm, [3], is correct and terminates.
**Generic Push-Relabel Analysis Lemma and Proof**

**Lemma 1:** If $f$ is a preflow, $d$ is any valid labeling for $f$, and $v$ is any active vertex, then either a push or a relabel operation is applicable to $v$. 

From [3].
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 1:** If $f$ is a preflow, $d$ is any valid labeling for $f$, and $v$ is any active vertex, then either a push or a relabel operation is applicable to $v$.

**Pf:**

Valid labeling states that $d(v) \leq d(w) + 1$ for any residual edge $(v, w)$.

If a push is not applicable to the active vertex $v$, then $d(v) < d(w) + 1$ for every residual edge $(v, w) \in E_f$.

By the integrality of valid labelings, $d(v) \leq d(w)$ for all residual edges $(v, w)$ and a relabeling is applicable to $v$.

From [3].
G E N E R I C  P U S H - R E L A B E L  A N A L Y S I S
L E M M A  A N D  P R O O F

Lemma 2: The algorithm maintains the condition that \( d \) is a valid labeling.
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 2:** The algorithm maintains the condition that $d$ is a valid labeling.

**Pf:**

Sorry. Exercise question.
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 3:** If a preflow, $f$, and a valid labeling, $d$, for $f$ exists then there is no augmenting path from $s$ to $t$ in the residual graph $G_f$. 

*Presented by Kenzie MacNeil*

*Submitted to Prof. Anil Maheshwari for COMP 5703*

*From [3].*
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 3:** If a preflow, $f$, and a valid labeling, $d$, for $f$ exists then there is no augmenting path from $s$ to $t$ in the residual graph $G_f$.

**Pf:**

By contradiction, let us assume there is some augmenting path $v_0, v_1, ..., v_n$ in $G_f$ such that $s = v_0$ and $t = v_n$.

This augmenting path must have the following properties:
- $n < |V|$;
- $(v_i, v_{i+1}) \in E_f$ for $0 \leq i < n$; and
- $d(v_i) \leq d(v_{i+1}) + 1$ for $0 \leq i < n$ given $d$ is a valid labeling.

However, $d(s) \leq d(t) + n < |V|$ forms a contradiction given the $d(s) = |V|$ and $d(t) = 0$ of the valid labeling.

Therefore, there can be no augmenting path from $s$ to $t$ in the residual graph $G_f$. 

From [3].
**Generic Push-Relabel Analysis**

**Theorem 1:**
Suppose the generic algorithm terminates and all labels $d(v)$ for a preflow $f$ are finite at termination; then the preflow $f$ is a maximum flow and the algorithm is correct.

**Pf:**

If the algorithm terminates and all labels are finite, then all vertices in $V - \{s, t\}$ are not active.

This means all $v \in V - \{s, t\}$ have no excess flow, and with no excess the preflow $f$ obeys the conservation constraint.

This flow is also the maximum flow given there is no augmenting path in the residual graph, by Lemma 3 and the Max-Flow, Min-Cut theorem.
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 4**: If a vertex $v$ in preflow $f$ has positive excess, then there exists a path from $v$ to the source $s$ in the residual graph $G_f$.

From [3, 5].
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 4:** If a vertex $v$ in preflow $f$ has positive excess, then there exists a path from $v$ to the source $s$ in the residual graph $G_f$.

**Pf:**
Let $A$ be the set of all vertices such that there exists a path from $v \in A$ to the source $s$ in $G_f$.
Let $B = V - A$ such that there no path in $G_f$ from $w \in B$ to $s$.

Given these sets, it is impossible for an edge $(u, w)$, where $u \in A$ and $w \in B$, to have a preflow larger than zero because it would cause a residual edge $(w, u)$ in $G_f$.
This $(w, u)$ would create a path from $w$ to $s$ in $G_f$ which violates the set definition.

Thus, $f(u, w) \leq 0$ for $u \in A$ and $w \in B$.

* This proof continues onto the next slide.
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 4**: If a vertex \( v \) in preflow \( f \) has positive excess, then there exists a path from \( v \) to the source \( s \) in the residual graph \( G_f \).

**Pf (cont’d):**
Consider the sum excess flow in the set \( B \):

\[
\sum_{w \in B} e_f(w) = \sum_{v \in V, w \in B} f(v, w)
\]

\[
\sum_{w \in B} e_f(w) = \sum_{u \in A, w \in B} f(u, w) + \sum_{y, w \in B} f(y, w)
\]

Recall that \( \sum_{y, w \in B} f(y, w) = 0 \) according to the skew symmetry and that \( f(u, w) \leq 0 \) for \( u \in A \) and \( w \in B \), and that by definition, we cannot have negative flow excess, \( e_f(v) \geq 0 \) for all \( v \in V \).

Thus, \( \sum_{w \in B} e_f(w) = \sum_{u \in A, w \in B} f(u, w) \leq 0 \).

Therefore, \( e_f(w) = 0 \) for all \( w \in B \) which means all vertices with excess must exist in set \( A \) and have a path for \( v, v \in A \), to the source \( s \) in the residual graph \( G_f \).
**Generic Push-Relabel Analysis Lemma and Proof**

**Lemma 5:** The label $d(v)$ never decreases for any vertex $v$ and, by extension, the relabeling operation on $v$ can only increase $d(v)$.

From [3].
Generic Push-Relabel Analysis
Lemma and Proof

Lemma 5: The label $d(v)$ never decreases for any vertex $v$ and, by extension, the relabeling operation on $v$ can only increase $d(v)$.

Pf:
Suppose there is some relabel operation performed on $v$. The relabel operation requires $d(v) \leq d(w)$ for all $w$ such that $(v, w) \in E_f$ which implies that the operation of relabeling must result in

$$d(v) < \min\{d(w) + 1 \mid (v, w) \in E_f\}.$$ 

This also proves that $d(v)$ never decreases since the relabel operation is the only part of the algorithm which modifies the $d(v)$ value.

From [3].
**Generic Push-Relabel Analysis Lemma and Proof**

**Lemma 6:** The $d(v)$ value of all vertices, $v \in V$, are $d(v) \leq 2|V| - 1$ throughout any execution of the algorithm.
Generic Push-Relabel Analysis
Lemma and Proof

**Lemma 6:** The $d(v)$ value of all vertices, $v \in V$, are $d(v) \leq 2|V| - 1$ throughout any execution of the algorithm.

**Pf:**
The source, $d(s) = |V|$, and sink, $d(t) = 0$, label values do not change after initialization.

Suppose there is some active vertex $v$. If $v$ is active, then it must have $e_f(v) > 0$ which means, according to Lemma 4, that there exists some path from $v$ to $s$ in $G_f$.

* This proof continues onto the next slide.
Generic Push-Relabel Analysis
Lemma and Proof

Lemma 6: The \( d(v) \) value of all vertices, \( v \in V \), are \( d(v) \leq 2|V| - 1 \) throughout any execution of the algorithm.

Pf (cont’d):
Let \( v_0, v_1, ..., v_n \) be a path in \( G_f \) such that \( v_0 \) is the active vertex \( v \), \( v_n \) is the source and \( n \leq |V| - 1 \).
Recall that \( d \) is a valid labeling which means \( d(v_i) \leq d(v_{i+1}) + 1 \) for \( (v_i, v_{i+1}) \in E_f \).

The result is that the labeling in the path has the following relationship \( d(v) \leq d(s) + n \leq d(s) + (|V| - 1) \) which can be simplified to \( |V| + (|V| - 1) \) given the source condition.

From [3].
**Generic Push-Relabel Analysis**  
**Lemma and Proof**

**Lemma 7:** The number of relabeling operations during any execution of the algorithm is at most $2|V| - 1$ per vertex and at most $(2|V| - 1)(|V| - 2) < 2|V|^2$ overall.
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 7:** The number of relabeling operations during any execution of the algorithm is at most $2|V| - 1$ per vertex and at most $(2|V| - 1)(|V| - 2) < 2|V|^2$ overall.

**Pf:**
Recall that the relabel operation only applies to $v \in V - \{s,t\}$.

The label value of a vertex $v \in V - \{s,t\}$ is initialized to 0 and according to Lemma 6 the maximum value is $2|V| - 1$.

Therefore, the relabel operation can be performed at most $2|V| - 1$ times for $v \in V - \{s,t\}$ which simplifies to $(2|V| - 1)(|V| - 2)$.
**Generic Push-Relabel Analysis Lemma and Proof**

**Lemma 8:** The number of saturating push operations during any execution of the algorithm is at most $2 |V| |E|$. 

From [3].
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 8:** The number of saturating push operations during any execution of the algorithm is at most $2 |V| |E|$.  

**Pf:**  
Consider a saturating push from $v$ to $w$ and then from $w$ to $v$ given any pair of vertices $v, w \in V$, $(v, w) \in E$ and $(w, v) \in E$.  

In order to perform the first push from $v$ to $w$ the labeling must be $d(v) = d(w) + 1$. Thus, $d(w)$ must increase by at least 2 in order to push flow back to $v$.  

* This proof continues onto the next slide.
**Generic Push-Relabel Analysis**

**Lemma and Proof**

**Lemma 8:** The number of saturating push operations during any execution of the algorithm is at most $2 |V| |E|$. 

**Pf (cont’d):** 
The first push between $v$ and $w$ can only occur when $d(v) + d(w) \geq 1$, and it can be determined that the last push must occur when $d(v) + d(w) \leq 4 |V| - 3$ given Lemma 6 that for all vertex, $v \in V$, $d(v) \leq 2 |V| - 1$. 

Thus, the maximum total number of saturating pushes along an edge is $2 |V| - 1$; making the maximum total during the execution of the algorithm at most:

$$(2 |V| - 1) |E| < 2 |V| |E|.$$ 

From [3].
Lemma 9: The number of non-saturating push operations during any execution of the algorithm is at most $4 |V|^2 |E|$. From [3].
**Generic Push-Relabel Analysis Lemma and Proof**

**Lemma 9:** The number of non-saturating push operations during any execution of the algorithm is at most $4 \cdot |V| \cdot 2 \cdot |E|$. 

**Pf:**
Let $\Phi = \sum_v d(v)$ where $v$ is an active vertex. This means that $\Phi$ starts and terminates the algorithm with a value of 0.

The total increase to $\Phi$ for the relabel operation during the execution of the algorithm is at most $(2 \cdot |V| - 1) \cdot (|V| - 2)$, given Lemma 7.

A saturating push operation will increase $\Phi$ by at most the maximum label value. Thus, the total increase to $\Phi$ for saturating pushes during the execution of the algorithm is at most $(2 \cdot |V| - 1) \times 2 \cdot |V| \cdot |E|$, given Lemma 6 and 8.

* This proof continues onto the next slide.
**Generic Push-Relabel Analysis Lemma and Proof**

**Lemma 9:** The number of non-saturating push operations during any execution of the algorithm is at most $4|V|^2|E|$.

**Pf (cont’d):**
A non-saturating push operation will decrease $\Phi$ by at least 1 since the vertex pushing the flow cannot contain any remaining excess and is no longer considered active.

Thus, the total decrease to $\Phi$, which is caused by non-saturating push operations, must match with the total increase in $\Phi$ which is at most:

$$ (2|V| - 1)(|V| - 2) + (2|V| - 1)(2|V||E|) \leq 4|V|^2|E| $$

From [3].
**Generic Push-Relabel Analysis**

**Theorem 2:**
The generic algorithm terminates after $O(|V|^2|E|)$ basic operations.

**Pf:**
The generic algorithm can perform one of three basic operations: relabel, saturating push or non-saturating push.

The number of times these operations can be performed have been proven to be finite according to lemma 7, 8 and 9 with the non-saturating push operation being the bottleneck at $O(|V|^2|E|)$. From [3].
**Generic Push-Relabel Analysis**

- Proved that all labels $d(v)$ for a preflow $f$ are finite at termination; and
- The generic algorithm is correct because preflow $f$ is a maximum flow at termination making the generic algorithm correct.

- Proved that the generic algorithm terminates after $O(|V|^2 |E|)$ time.
Conclusion
The generic Push-Relabel algorithm outlined in [3] can determine the maximum flow of a network flow graph in $O(|V|^2 |E|)$ time.

This assumes a reasonable sequential implementation of the order of the push and relabel operations.
**Push-Relabel Algorithm Alternative Implementations**

- Goldberg and Tarjan provided a sequential implementation of the Push-Relabel algorithm in [3] with a time bound of $O(|V|^3)$.

- Cheriyan and Maheshwari provided a different sequential implementation of the Push-Relabel algorithm in [7] with a time bound of $O(|V|^2 \sqrt{|E|})$.

- Goldberg and Tarjan also provided another implementation of Push-Relabel algorithm in [3] which used dynamic trees and had a time bound of $O(|V| |E| \log \frac{|V|^2}{|E|})$.

- Dynamic trees, or Link/Cut trees, are a data structure created by Sleator and Tarjan, [8, 9, 10].
QUESTIONS OR COMMENTS?

Criticism...?
BIBLIOGRAPHY


