1.1 Push-Relabel Algorithm

The Push-Relabel, or Preflow-Push, Maximum-Flow algorithm was originally developed by Andrew V. Goldberg and Robert E. Tarjan in the mid-to-late 1980s, [2, 3]. This Push-Relabel algorithm is unlike the previous Ford-Fulkerson and Edmonds-Karp algorithms which use the concept of an augmenting path to determine the maximum flow. This algorithm instead uses a concept of a preflow, originally developed by A. V. Karzanov [4], in order to determine the maximum flow on a flow network graph \( G = (V, E) \), [3].

1.1.1 Generic Push-Relabel

The Push-Relabel algorithm, as described in [3], works by manipulating a preflow of the original flow network. In [3], Goldberg and Tarjan outline a generic version of their Push-Relabel algorithm. This generic algorithm defines the general operations required but does not specify the operation ordering or discusses implementation details.

The algorithm works at converting a preflow, \( f \), to a flow by pushing local flow excess starting at the source, \( s \), along the vertices towards the sink, \( t \). To accomplish this, the algorithm maintains a compatible vertex labeling function, \( d \), to the preflow \( f \) in order to determine where to push the flow excess. The algorithm terminates once the preflow becomes a flow. This final flow produced at termination also turns out to be the maximum flow for the graph.

Non-Negative Constraint is a weakened form of the flow conservation constraint which states
that the flow into \( v \in V - \{s\} \) must be greater or equal to the total flow out of \( v \).

\[
\sum_{u \in V} f(u, v) \geq 0, \quad \forall v \in V - \{s\}
\]

**Flow Excess** is the net flow into \( v \) where \( v \in V \) for some preflow, \( f \).

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

![Figure 1.1: Simple example of a preflow at vertex v.](image)

**Active Vertex** is a vertex \( v \) which satisfies all of the properties: \( v \in V - \{s, t\} \), \( e_f(v) > 0 \) and \( d(v) < \infty \).

The Push-Relabel algorithm also uses the concepts of a residual graph which was previously defined in the previous Ford-Fulkerson algorithm. The algorithm defines the **residual capacity** for a preflow as \( r_f(v, w) = c(v, w) - f(v, w) \) and a **residual graph**, \( G_f \), for a preflow \( f \) as \( G_f = (V, E_f) \) where \( E_f \) contains the set of residual edges, \( r_f(v, w) > 0 \), such that \( E_f = \{(v, w)|r_f(v, w) > 0\} \).

The algorithm uses a push operation, outlined in Function Push, to move flow excess from one vertex to another. This transfer of excess can be performed across a vertex pair \((v, w) \in E_f\) if \( v \) is an active vertex with a positive residual capacity, \( r_f(v, w) > 0 \) and a label distance of \( d(v) = d(w) + 1 \). This will allow the algorithm to move delta excess flow, \( \delta = \min(e_f(v), r_f(v, w)) \), from \( v \) to \( w \) by adding \( \delta \) to \( f(v, w) \) and subtracting \( \delta \) from \( f(w, v) \). A push is considered saturating if no more flow can be move over the edge, \( \delta = r_f(v, w) \); otherwise the push is considered non-saturated.

**Function Push**

/* Pseudocode adapted from [3] */

**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 \( d(v) = d(w) + 1 \)

\[
\begin{align*}
\delta & \leftarrow \min(e_f(v), r_f(v, w)) \\
f(v, w) & \leftarrow f(v, w) + \delta \\
f(w, v) & \leftarrow f(w, v) - \delta \\
e_f(v) & \leftarrow e_f(v) - \delta \\
e_f(w) & \leftarrow e_f(w) + \delta \\
\end{align*}
\]

return \( f \);

The algorithm 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, or height, such that $0 \leq d(v) \leq \infty$, $d(s) = |V|$, $d(t) = 0$ and $d(v) \leq d(w) + 1$ for every residual edge $(v, w) \in E_f$. In the algorithm, the labeling for the source and the sink are initialized with the following values $d(s) = |V|$ and $d(t) = 0$. These are referred to as the source condition and sink condition respectively, [5].

The intent of the labeling is to organize the vertices in the residual graph such that the vertices in upper bound of $d$ are closer to the source, $s$, and the vertices in the lower bound are closer to the sink, $t$. According to the algorithm the push can only occur from vertices with higher label values to vertices with smaller label values. This function is sometimes compared to the physical intuition that liquids naturally flow downhill, [5].

Furthermore, a labeling $d$ and a preflow $f$ are said to be compatible if $d$ adheres to source condition, sink condition and $d(v) \leq d(w) + 1$, $\forall (v, w) \in E_f$. The algorithm repeatedly performs either the relabel operation, outlined in Function Relabel, or the push operation so long as there is an active vertex in $G_f$. This is proven in Lemma 1.1.1 from [3].

**Lemma 1.1.1** 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 $v$.

**Proof.** The definition of a 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 $v$.

```plaintext
Function Relabel
/* Pseudocode adapted from [3] */

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 $d(v) \leq d(w)$

/* If minimum is not over an empty set of residual edges. */
if \{(v, w) \in E_f\} \neq \emptyset then
    $d(v) \leftarrow \min\{d(w) + 1\} | (v, w) \in E_f$
else
    $d(v) \leftarrow \infty$
end
return $d$;
```

Using the push and relabel operations the algorithm maintains a preflow $f$ and a labeling $d$ which is compatible with $f$ in order to produce a flow. The full algorithm, Algorithm 1.1, starts by initializing the preflow of all edges on the residual graph to zero, initializing the labeling and then pushing the maximum flow along all $f(s, v), v \in V$.

Once the initialization is complete, the algorithm repeatedly performs either the push or relabel operations against active vertices until no applicable operation can performed. The algorithm terminates when there are no more active vertices. See Example 1.1.2 for a walkthrough of the algorithm’s execution.
Algorithm 1.1: Push-Relabel
/* Pseudocode adapted from [3] */

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

Output: A maximum flow $f$

/* Initialize preflow */
forall the $(v, w) \in (V - \{s\}) \times (V - \{s\})$ do
  $f(v, w) \leftarrow 0$
  $f(w, v) \leftarrow 0$
end
forall the $v \in V$ do
  $f(s, v) \leftarrow r_f(s, v)$
  $f(v, s) \leftarrow -r_f(s, v)$
end
/* Initialize labels and excesses */
$d(s) \leftarrow |V|$
forall the $v \in V - \{s\}$ do
  $d(v) \leftarrow 0$
  $e_f(v) \leftarrow f(s, v)$
end
/* Loop while there exists an active vertex */
while $\exists v \in V - \{s, t\}$ with either applicable PUSH or RELABEL operation do
  Perform either a PUSH() or a RELABEL().
end
return $f$;
Example 1.1.2 The following is a sample execution of the generic Push-Relabel algorithm as defined in [3] on the network flow graph, \(G = (V, E)\), from Figure 4.1 in [6]. In Figure 1.2, we initialize the graph by setting the preflow to 0, initializing the labeling and pushing flow from the source, \(s\).

![Figure 1.2: Push-Relabel residual graph, \(G_f\): Initialization of labels and excess.](image1)

In Figure 1.3, the vertex \(A\) is relabeled in order to push flow towards the sink, \(t\). This excess is consumed by \(B\) and \(D\) in two subsequent saturating pushes; still leaving \(A\) with excess. In order to remove its excess flow, vertex \(A\), in 1.4, once again relabels in order to push its flow along its last remaining positive residual edge and pushes the excess back to \(s\). The vertex \(A\) is then removed from the set of active vertices.

![Figure 1.3: Push-Relabel \(G_f\): Relabel \(a\) then push excess along \((a, b), (a, d) \in E_f\).](image2)

In Figures 1.5, 1.6 and 1.7, a vertex is first relabeled then pushes its excess towards \(t\). These operations leave \(B\) as the sole active vertex who cannot push flow towards the sink. In Figures 1.8 and 1.9, the \(B\) vertex is relabeled and the remaining excess in the residual graph is pushed back along \(A\) to the source. The resulting residual graph represents the max flow for Figure 4.1 in [6].

The following theorems, lemmas and proofs found in [3] and [5] are used to prove that the generic algorithm, Algorithm 1.1, is correct and terminates.

**Lemma 1.1.3** The algorithm maintains the condition that \(d\) is a valid labeling.
Lemma 1.1.4 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 $G_f$.

**Proof.** 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 that $d$ is a valid labeling. However, $d(s) \leq d(t) + n < |V|$ forms a contradiction given the source condition, $d(s) = |V|$, and sink condition, $d(t) = 0$, of the valid labeling. Therefore, there can be no augmenting path from $s$ to $t$ in the residual graph $G_f$. \hfill \blacksquare

Given these lemmas we can now outline a theorem for the correctness of the generic Push-Relabel algorithm.

**Theorem 1.1.5** 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.

**Proof.** 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 and can be considered a flow. This flow is also the maximum flow given
there is no augmenting path in the residual graph, by lemma 1.1.4 and max-flow min-cut theorem.

Now we must prove that the generic algorithm terminates with a finite labeling $d$ for a preflow $f$ for all executions of the algorithm.

**Lemma 1.1.6** 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 $G_f$.

**Proof.** 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 $\bar{A} = V - A$ such that there no path in $G_f$ from $w \in \bar{A}$ to $s$. Given these sets, it is impossible for an edge $(u, w)$, where $u \in A$ and $w \in \bar{A}$, to have a preflow larger than zero, $f(u, w) > 0$, 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 $w \in \bar{A}$.

Consider the sum excess flow in the set $\bar{A}$.

$$\sum_{w \in \bar{A}} e_f(w) = \sum_{v \in V, w \in A} f(v, w)$$

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

Therefore, \( e_f(w) = 0 \) for all \( w \in \bar{A} \) 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 \( G_f \).

**Lemma 1.1.7** The label \( d(v) \) never decreases for any vertex \( v \) and, by extension, the relabeling operation on \( v \) can only increase \( d(v) \).

**Proof.** 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 \( d(v) < \min\{d(w) + 1 | (v, w) \in E_f \} \). 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.

**Lemma 1.1.8** The \( d(v) \) value of all vertices, \( v \in V \), are \( d(v) \leq 2|V| - 1 \) throughout any execution of the algorithm.

**Proof.** 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 1.1.6, that there exists some path from \( v \) to \( s \) in \( G_f \). 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. 

Therefore, lemma 1.1.8 implies there is a finite height to the valid labeling values during execution of the algorithm. This will help bound the number of relabel and push operations within the algorithm.

**Lemma 1.1.9** 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.

**Proof.** 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 1.1.8 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 \((2|V| - 1)(|V| - 2)\). 

**Lemma 1.1.10** The number of saturating push operations during any execution of the algorithm is at most \( 2|V||E| \).

**Proof.** 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 \( v \).

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 1.1.8 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 \((2|V| - 1)|E| < 2|V||E|\).

**Lemma 1.1.11** The number of non-saturating push operations during any execution of the algorithm is at most \( 4|V|^2|E| \).

**Proof.** 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 \((2|V| - 1)(|V| - 2)\), given lemma 1.1.9.

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 \((2|V| - 1) \times 2|V||E|\), given lemma 1.1.8 and 1.1.10.

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

Given these lemmas we can now outline a theorem for the termination of the generic Push-Relabel algorithm.
Theorem 1.1.12  The generic algorithm terminates after $O(|V|^2|E|)$ basic operations.

Proof. During its execution 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 1.1.9, lemma 1.1.10 and lemma 1.1.11 with the non-saturating push operation being the bottleneck at $O(|V|^2|E|)$.

Therefore, the generic Push-Relabel algorithm outlined in [3] can determine the maximum flow of a network flow graph in $O(|V|^2|E|)$ time regardless of the order in which the push and relabel operations are applied.

However, there have been several proposed implementations of this generic Push-Relabel algorithm which outline data structures and/or stricter rules for operations selection. Goldberg and Tarjan provided one such sequential implementation in [3] which improved the order of push and relabel operation. They proved that their sequential implementation of the Push-Relabel algorithm has a time bound of $O(|V|^3)$. Likewise, [1] provided another sequential implementation of the Push-Relabel algorithm which focused on managing the push relabel operations. They proved that their Push-Relabel implementation has a time bound of $O(|V|^2\sqrt{|E|})$. Finally, [3] also provided an implementation of Push-Relabel algorithm using the dynamic tree, or link/cut tree, data structures of Sleator and Tarjan, [7, 8, 9]. This implementation was proved to have a time bound of $O(|V||E|\log \frac{|V|^2}{|E|})$.

1.2 Exercises

1.1 Given Goldberg and Tarjan’s generic Push-Relabel algorithm, prove that the algorithm maintains the condition that $d$ is a valid labeling.

1.2 Design a maximum flow algorithm by modifying Goldberg and Tarjan’s generic Push-Relabel algorithm. This new algorithm must terminate in $O(|V|^3)$ time.

Hint: Try to reduce the number of nonsaturating pushes by using basic data structures.
Bibliography


