Lecture Notes CS:5350 Design and Analysis of Algorithms
Lecture: Sep 17, 2019
Scribe: Zhishuai Guo

1 Analysis of the Edmonds-Karp "fewest pipes" heuristic

In last lecture, we have learned the Edmonds-Karp "fewest pipes" heuristic, which is: in each iteration, an augmenting path is chosen such that it has fewest edges.

Theorem 1 *This heuristic runs in $O(m^2n)$ time, even if the capacities are not integral.*

To prove this theorem, we will show that an edge in residual graph can disappear and reappear at most $n/2$ times.

Let $f_0, f_1, ...$ be the sequence of flows constructed by the algorithm. Note that f_0 in the initial flow which is an all-zero flow. Let G_i be a shorthand for G_{f_i}. Let $level_i(v)$ be the shortest path distance from s to v in G_i (we only consider the number of edges in the path, i.e., the edges are unweighted).

Lemma 2 *For all $v \in V$ and all $i \geq 1$, $level_i(v) \geq level_{i-1}(v)$.*

Proof Sketch. There are two special kinds of vertices: (1) $v = s$, then $level_i(v) = 0$ for all i; (2) v is unreachable for s in G_i, then $level_i(v) = \infty$. In both cases, $level_i(v) \geq level_{i-1}(v)$.

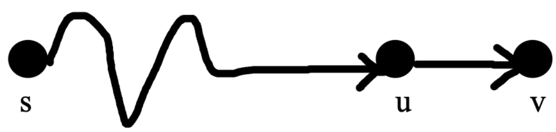

Figure 1: A shortest path from s to v in G_i, u is the vertex previous to v in this path.

Figure 1 shows a shortest path from s to v in G_i, and u is the previous vertex to v. Therefore, $level_i(u) + 1 = level_i(v)$ since we know that a shortest path to v is also a shortest path to every vertex earlier in the path.

Induction hypothesis (IH): $level_{i-1}(u) \leq level_i(u)$ for all u such that $level_i(u) < level_i(v)$. (The base case is for vertex s.)

There are two possible cases: (1) $u \to v$ in G_i; (2) $u \to v$ not in G_{i-1}.

(1) $u \to v$ in G_i:

$$\begin{aligned} level_{i-1}(v) &\leq level_{i-1}(u) + 1 && \text{since it is possible to reach } v \text{ via } u \text{ in } G_{i-1}. \\ &\leq level_i(u) + 1 && \text{by induction hypothesis} \\ &= level_i(v) - 1 + 1 && \text{because } u \to v \text{ is in the shortest path in } G_i \\ &= level_i(v) \end{aligned} \tag{1}$$

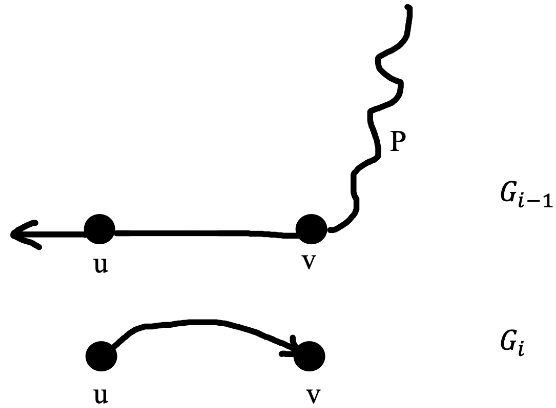

Figure 2: Illustration of case (2). When $u \to v$ is not in G_{i-1}, but in G_i. So the augmenting path P includes $v \to u$

(2) $u \to v$ not in G_{i-1}: We want to know why $u \to v$ is in G_i but not in G_{i-1}. There can be two scenarios. The first one is that $u \to v$ is an edge in the original graph G. Then $u \to v$ disappears in G_{i-1} because it is saturated in flow f_{i-1}. Then by the shortest pipe heuristic, a path P has been picked up for the flow f_{i-1} as an augmenting path such that P is a shortest $s \rightsquigarrow t$ path in G_{i-1}. $u \to v$ reappears in G_i because that $v \to u$ is in P (i.e., some flow has been pushed back from v to u, and hence, $u \to v$ is not saturated anymore).

The second scenario is that $v \to u$ is an edge in G. Then $u \to v$ is not in G_{i-1} because there is no flow from v to u in f_{i-1}. Again, by the shortest pipe heuristic, a path P has been picked up for the flow f_{i-1} as an augmenting path such that P is a shortest $s \rightsquigarrow t$ path in G_{i-1}. $u \to v$ appears in G_i because that $v \to u$ is in P, i.e., there is some flow from v to u in f_i.

Therefore, in both scenarios, $v \to u$ is included in the augmenting P which is a shortest $s \rightsquigarrow t$ path in G_{i-1}.

$$\begin{aligned}
level_{i-1}(v) &= level_{i-1}(u) - 1, && \text{since } v \to u \text{ is in } P \text{ and } P \text{ is a shortest} s \rightsquigarrow t \text{ path in } G_{i-1} \\
&\leq level_i(u) - 1, && \text{by induction hypothesis} \\
&= level_i(v) - 1 - 1, && \text{since } u \to v \text{ is in shortest path in } G_i \\
&= level_i(v) - 2
\end{aligned} \tag{2}$$

■

Lemma 3 *An edge $u \to v$ can reappear in the residual graph sequence $G_0, G_1, ...$ at most $\frac{n}{2}$ times.*

Proof Sketch.

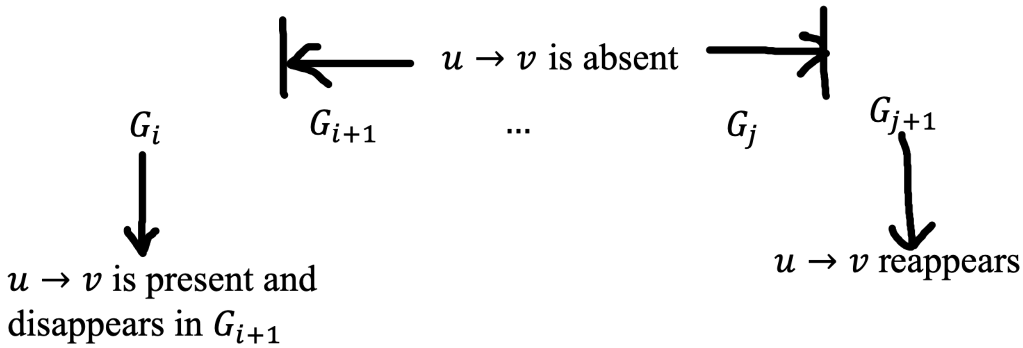

Figure 3: Illustration of the disappear once and reappear once of an edge $u \to v$.

Since $u \to v$ is in G_i but disappears in G_{i+1}, we know that $u \to v$ is in the augmenting path P_i which is a shortest $s \rightsquigarrow t$ path in G_i. To see this, we can again consider two scenarios. The first one is that $u \to v$ is in the original graph G. $u \to v$ exists in G_i but not in G_{i+1}. Then we know it is not saturated in f_i but saturated in f_{i+1}. Thus, $u \to v$ is in the augmenting path P_i which is a shortest $s \rightsquigarrow t$ path in G_i (i.e., P_i pushes more flow from u to v in to make it saturated). The second scenario is that $v \to u$ is in the original graph G. Then we know there is some flow from v to u in f_i and no flow from v to u in G_{i+1} (i.e., all the flow from v to u is pushed back by P_i). Thus, $u \to v$ is in the augmenting path P_i which is a shortest $s \rightsquigarrow t$ path in G_i. Thus, in both scenarios, we have $level_i(v) = level_i(u) + 1$.

Since $u \to v$ is not in G_j but in G_{j+1}, we know that $v \to u$ is in the augmenting path P_j which is a shortest $s \rightsquigarrow t$ path in G_j. The analysis of this is similar to the above. Thus we have $level_j(u) = level_j(v) + 1$.

$$\begin{aligned} level_j(v) &= level_j(u) - 1 \\ &\geq level_i(u) - 1, \qquad \text{by Lemma 2} \\ &= level_i(v) - 2 \end{aligned} \tag{3}$$

Level of v increase by at least 2 when $u \to v$ disappears and reappear. Thus, $u \to v$ can reappear at most $\frac{n}{2}$ times since the level of v, which is the shortest length from s to v, cannot be larger than the number of vertices (The level of v can achieve this number only when the path go through all the other vertices and then finally reaches v). Because each edge disappears at most $n/2$ times, there are at most $mn/2$ edge disappearances overall. We also know that in each iteration, the value of the augmenting path is taken to saturate the edge that requires least flow to be saturated (the minimum residual capacity of the edges in the augmenting path). So we saturated at least one edge in each iteration. That means, at least one edge disappears on each iteration, so the algorithm must halt after at most $mn/2$ iterations. Finally, looking for the shortest path in each iteration requires $O(m)$ time, so the overall algorithm runs in $O(m^2n)$ time. ∎

2 Introduction to LP duality

The MaxFlow-MinCut theorem is a special case of an important duality result in LP. In MaxFlow-MinCut problem, the value of any arbitrary feasible flow is always upper bounded by an arbitrary capacity of a cut. And the maximum value of the flow is equal to the minimum capacity of the cut. Generally, a primal objective value is always upper bounded by a dual objective value (weak duality). And the maximum primal objective value is equal to the the minimum dual objective value (strong duality).

Example:

$$
\begin{aligned}
\max_{x_1,x_2,x_3,x_4} \quad & x_1 + 2x_2 + x_3 + x_4 \\
& x_1 + 2x_2 + x_3 \leq 2 && (i) \\
& x_2 + x_4 \leq 1 && (ii) \\
& x_1 + 2x_3 \leq 1 && (iii) \\
& x_1, x_2, x_3, x_4 \geq 0
\end{aligned}
\tag{4}
$$

Suppose we are given a solution is given $x_1 = 1, x_2 = \frac{1}{2}, x_3 = 0, x_4 = \frac{1}{2}$. The corresponding objective function value is 2.5. Now we want to check if the this is optimal without solving LP.

(1) We consider $(i) + (ii)$, then we have a constraint: $x_1 + 3x_2 + x_3 + x_4 \leq 3$. Since we also have $x_1 + 2x_2 + x_3 + x_4 \leq x_1 + 3x_2 + x_3 + x_4$ for all $x_1, x_2, x_3, x_4 \geq 0$, we know 3 is an upper bound of the objective value.

(2) We consider $\frac{1}{2}(i) + (ii) + \frac{1}{2}(iii)$, then we have a constraint: $x_1 + 2x_2 + 1.5x_3 + x_4 \leq 2.5$. Since we also have $x_1 + 2x_2 + x_3 + x_4 \leq x_1 + 2x_2 + 1.5x_3 + x_4$, we know that 2.5 is an upper bound of the objective value. Thus the given solution set $x_1 = 1, x_2 = \frac{1}{2}, x_3 = 0, x_4 = \frac{1}{2}$ can achieve the objective value of 2.5, and we know this set of solution is optimal.

Thus, to check whether a given set of solution is optimal or not, we want to find the tightest bound. That is, we need to find a set of multipliers (y) for the constraints to get the best upper bound.