Residual Graph in Maximum Flow

Asked 6 years, 5 months ago Modified 5 years, 3 months ago Viewed 49k times

20

14

I am reading about the Maximum Flow Problem here. I could not understand the intuition behind the Residual Graph. Why are we considering back edges while calculating the flow?

Can anyone help me understand the concept of Residual Graph?

How does the Algorithm change in Undirected Graphs?

algorithms graphs network-flow

Share Cite Edit Follow Flag

edited Mar 14, 2017 at 2:38
Mario Cervera
3,474 2 17 22

asked Mar 28, 2016 at 5:22
csds
201 1 2 3

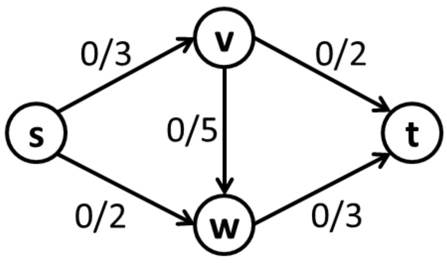
2 Answers

Sorted by: Highest score (default)

37

The intuition behing the residual graph in the Maximum flow problem is very well presented in this lecture. The explanation goes as follows.

Suppose that we are trying to solve the maximum flow problem for the following network G (where each label f_e/c_e denotes both the flow f_e pushed through an edge e and the capacity c_e of this edge):



One possible greedy approach is the following:

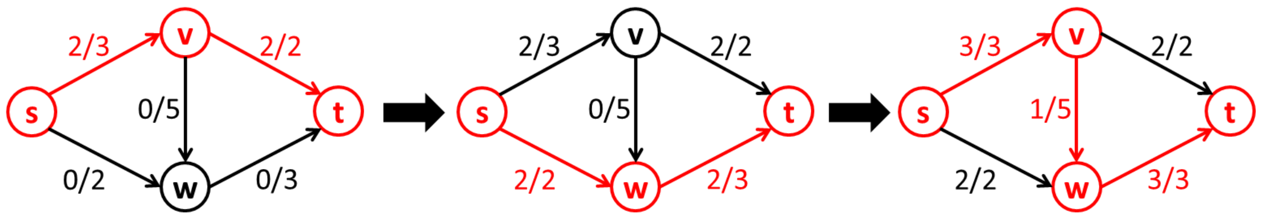
1. Pick an arbitrary augmenting path P that goes from the source vertex s to the sink vertex t such that $\forall e \; (e \in P \rightarrow f_e < c_e)$; that is, all of the edges in P have available capacity.
2. Push the maximum possible flow Δ through this path. The value of Δ is determined by the *bottleneck* of P; that is, the edge with minimum available capacity. Formally,

$\Delta = \min_{e \in P}(c_e - f_e)$.

3. Go to step 1 until no augmenting paths exist.

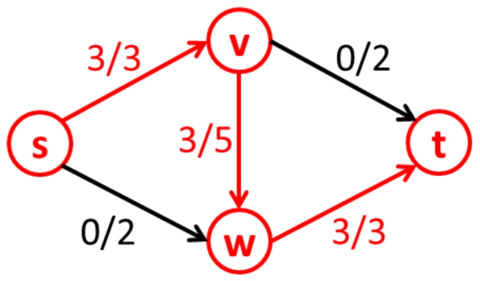
That is, find a path with available capacity, send flow along that path, and repeat.

In G, one possible execution of the above heuristic finds three augmenting paths P_1, P_2, and P_3, in this order. These paths push 2, 2, and 1 units of flow, respectively, for a total flow of 5:

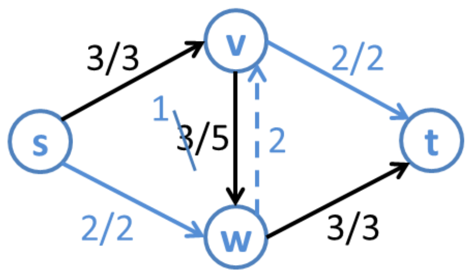


P_1 = [s, v, t] P_2 = [s, w, t] P_3 = [s, v, w, t]

Choosing paths in this order leads to an optimal solution; however, what happens if we select P_3 first (i.e., before P_1 and P_2)?



We get what is called a *blocking flow*: no more augmenting paths exist. In this case, the total flow is 3, which is not optimal. This issue can be resolved by allowing *undo* operations (i.e., by allowing flow to be sent in reverse, undoing work of previous iterations): simply push 2 units of flow backwards from vertex w to vertex v like this:

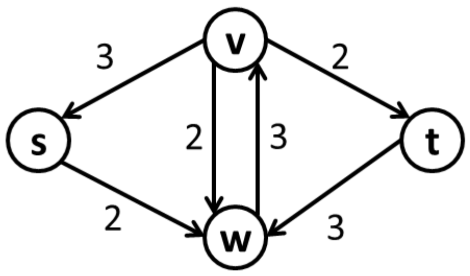


Encoding these allowed undo operations is the main goal of the **residual graph**.

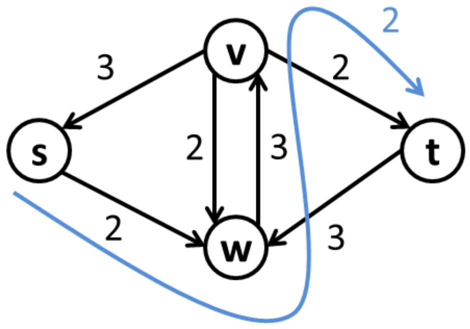
A residual graph R of a network G has the same set of vertices as G and includes, for each edge $e = (u, v) \in G$:

- A forward edge $e' = (u, v)$ with capacity $c_e - f_e$, if $c_e - f_e > 0$.
- A backward edge $e'' = (v, u)$ with capacity f_e, if $f_e > 0$.

For example, consider the residual graph R that is obtained after the first iteration of the greedy heuristic when the heuristic selects P_3 first (that is, when it gets the blocking flow):



Note that the *undo* operation that pushes 2 units of flow from w to v is encoded as a forward (augmenting) path from s to t in R:



In general:

> When an augmenting path P' is selected in the residual graph R:
>
> - Every edge in P' that corresponds to a forward edge in G increases the flow by using an edge with available capacity.
> - Every edge in P' that corresponds to an edge going backwards in G undoes flow that was pushed in the forward direction in the past.

This is the main idea behind the Ford–Fulkerson method.

The Ford–Fulkerson method proceeds in exactly the same way as the greedy approach described above, but it only stops when there are no more augmenting paths in the residual graph (not in the original network). The method is correct (i.e., it always computes a maximum flow) because the residual graph establishes the following **optimality condition**:

> Given a network G, a flow f is maximum in G if there is no $s - t$ path in the residual graph.

Share Cite Edit Follow Flag

edited May 22, 2017 at 16:39

answered Mar 13, 2017 at 23:58
Mario Cervera
3,474 2 17 22

Is there an example where paths are added in the order of shortest length as described in Edmonds-

Is there an example where paths are added in the order of shortest length as described in Edmonds Karp algorithm? In your counter example the first path is length 3 while a shorter (i.e. 2) path can be found and would be added first if we are doing Edmonds-Karp. – Roy Feb 25, 2018 at 6:20

You can simply make all $s-t$ paths in the original graph have length 3. To do so, split vertex v into two vertices v_1 and v_2. Then, split w into w_1 and w_2. Add also two edges (v_1, v_2) and (w_1, w_2) with capacity 2. The edge that originally went from v to w will now go from v_1 to w_2. We can obtain the same kind of blocking flow if we choose initially the path that contains the edge (v_1, w_2). – Mario Cervera Feb 26, 2018 at 16:56

Your example makes sense. We can always extend the graph on other edges in the cut to make the edge in question be on one of the shortest paths. – Roy Mar 1, 2018 at 20:04

4

The intuition behind the residual network is that it allows us to "cancel" an already assigned flow i.e. if we have already assigned 2 units of flow from A to B, then passing 1 unit of flow from B to A is interpreted as cancelling one unit of the original flow from A to B.

Share Cite Edit Follow Flag

answered Mar 28, 2016 at 7:22

Banach Tarski
1,168 7 19