Augmenting paths

Monday, November 7, 2016

This emendation to today's lecture provides some fixes to bugs in the AUGMENTINGPATH algorithm, as well as discussion of other possible implementations.

1. **Bug fixes.** The algorithm from class should be patched as follows.
   Lines 1-2 and 10-11 are new/changed.

AUGMENTINGPATH$(G = (V, E), s, t, \{c_e\}, f)$

Precondition: For each $e \in E$, input values c_e and $f(e)$ are ≥ 0. Also, s and $t \in V$.
Postcondition: Returns an augmenting path from s to t, or NULL if none exists.

1 **if** $e = (s, t) \in E$ and $c_e - f(e) > 0$:
2   **return** $[(s, t)]$ // base case of recursion
3 **for** each neighbor v of s:
4   **if** $(s, v) \in E$:
5         $\ell = c_{(s,v)} - f((s, v))$
6   **elseif** $(v, s) \in E$:
7         $\ell = f((v, s))$ // this is the amount of flow we can "push back"
8   **if** $\ell > 0$:
9   // then we can start our augmenting path from s to v
10         $G' = (V', E')$ where $V' = V \backslash \{s\}$ and $E' = E - \{\text{all edges involving } s\}$
11   temp = AUGMENTINGPATH$(G', v, t, \{c_e\}, f)$
12   **if** temp $\neq$ NULL:
13   **return** temp.prepend$((s, v))$
14 **return** NULL

The MAXFLOW algorithm remains the same as in class:

MAXFLOW$(G = (V, E), s, t, \{c_e\})$

Precondition: G is a directed edge, and $c_e \geq 0$ for all $e \in E$. Also, s and $t \in V$.
Postcondition: Returns a valid flow from s to t on G which is maximal.

1 initialize $f(e) = 0$ for all $e \in E$
2 $P =$ AUGMENTINGPATH$(G, s, t, \{c_e\}, f)$
3 **while** $P \neq$ NULL:
4     $\ell = min_{e \in P} \begin{cases} c_e - f(e) & \text{if } e \in E \\ f(e^{\text{reverse}}) & \text{if } e^{\text{reverse}} \in E \end{cases}$
5   **for** $e \in P$:
6   **if** $e \in E$:
7             $f(e) = f(e) + \ell$
8   **else if** $e^{\text{reverse}} \in E$:
9             $f(e^{\text{reverse}}) = f(e^{\text{reverse}}) - \ell$
10         $P =$ AUGMENTINGPATH$(G, s, t, \{c_e\}, f)$
11 **return** f

2. **Alternate design using the residual graph.**

As students observed in class, maxFlow does not explicitly use the residual graph. (Although the *idea* of the residual graph is hovering around, just off-screen.)

We can modify both maxFlow and augmentingPath to explicitly use the residual graph:

maxFlow2($G = (V, E), s, t, \{c_e\}$)

Precondition: G is a directed edge, and $c_e \geq 0$ for all $e \in E$. Also, s and $t \in V$.

Postcondition: Returns a valid flow from s to t on G which is maximal.

initialize $f(e) = 0$ for all $e \in E$
initialize $G_f = G$ and for each $e \in E$, $c'_e = c_e$
for $(u, v) \in E$: **if** $(v, u) \notin E$ then add (v, u) to G_f and define $c'_{(v,u)} = 0$
// Now G_f is the residual graph of G for flow f, with residual capacities $\{c'_e\}$.
$P =$ augmentingPath2($G_f, s, t, \{c'_e\}$)
while $P \neq$ null:
$$\ell = min_{e \in P} \begin{cases} c'_e & \text{if } e \in E \\ f(e^{\text{reverse}}) & \text{if } e^{\text{reverse}} \in E \end{cases}$$
    for $e = (u, v) \in P$:
        if $(u, v) \in E$:
            $f(e) = f(e) + \ell$ // increase the flow along this forward edge
            $c'_{(u,v)} = c'_{(u,v)} - \ell$ // residual forward capacity decreases
            $c'_{(v,u)} = c'_{(v,u)} + \ell$ // residual backward capacity increases
        elseif $(v, u) \in E$:
            $f((u, v)) = f((u, v)) - \ell$ // push back flow along (u, v)
            $c'_{(u,v)} = c'_{(u,v)} + \ell$ // residual forward capacity increases
            $c'_{(v,u)} = c'_{(v,u)} - \ell$ // residual backward capacity decreases
    $P =$ augmentingPath2($G_f, s, t, \{c'_e\}$)
return f

augmentingPath2($H = (V, E), s, t, \{c_e\}$)

Precondition: H is a residual graph (has all possible edges), $c_e \geq 0$ for all $e \in E$, and s and $t \in V$.

Postcondition: Returns an augmenting path from s to t, or null if none exists.

if $e = (s, t) \in E$ and $c_e \geq 0$:
    return $[(s, t)]$ // base case of recursion
for each neighbor v of s:
    if $c_{(s,v)} > 0$:
        $G' = (V', E')$ where $V' = V \backslash \{s\}$ and $E' = E - \{\text{all edges involving } s\}$
        temp = augmentingPath2($G', v, t, \{c_e\}$)
        if temp $\neq$ null
            return temp.prepend((s, v))
return null

3. **Alternate designs.** Other possible ideas for how to design AUGMENTINGPATH2:

Dijkstra: Use Dijkstra's algorithm on the residual graph G_f, traversing only edges e with $c_e > 0$. Keep track of "parent" nodes so that the augmenting path can be reconstructed.

BFS: Use BFS on the residual graph G_f starting at s, traversing only edges e with $c_e > 0$, to find whether t is reachable. Keep track of "parent" nodes in the BFS so that the augmenting path can be reconstructed.

DFS: Our implementation AUGMENTINGPATH2 above does the DFS version: starting at s, use DFS on the residual graph G_f starting at s, traversing only edges e with $c_e > 0$, to find whether t is reachable. Keep track of "parent" nodes (AUGMENTINGPATH2 does this in the call stack) so that the augmenting path can be reconstructed.