ECS 122A Homework 4

Hardy Jones
999397426
Professor Gysel
Fall 2014

1. We can find a possible augmenting path if we modify both of the given algorithms.

   For DFS, we do not need to look at each vertex when starting the algorithm. We already know we want to find a path from s to t, so we just need to check for paths from s.

   For DFS-VISIT, we should check if the current vertex is t. If it is, then we've constructed an st-path, and we can just return it. Otherwise, we continue as in the original algorithm. Each time we visit a new vertex, we add it to the end of the current path. If what was returned from our visitation is not an empty list, then we have an st-path, so we can return it. Otherwise, we continue searching for paths. If at the end of the search, we didn't find t, we return the empty list.

```
function AUGMENTING-PATH(G_f)
    for each v ∈ G_f do
        v.visited ← FALSE
    end for
    return ST-PATH(G_f, G_f.s, APPEND(NIL, G_f.s))
end function
```

```
function ST-PATH(G, v, l)
    v.visited ← TRUE
    if v == G.t then
        return l
    end if
    for each u ∈ G.Adj(v) do
        if u.visited == FALSE then
            path? ← ST-PATH(G, u, APPEND(l, u))
            if path? ≠ NIL then
                return path?
            end if
        end if
    end for
    return NIL
end function
```

2. We can solve this following the maximum bipartite matching problem and modifying it a bit.

   We create the two sets, $L = \{d_1, d_2, \ldots, d_n\}, R = \{p_1, p_2, \ldots, p_m\}$. Create the edges $E = \{(d_i, p_j) \mid i \in 1, 2, \ldots, n; j \in 1, 2, \ldots, m; p_j \in S_i\}$. Give each edge in E a weight of 1 to show that each doctor can treat one patient.

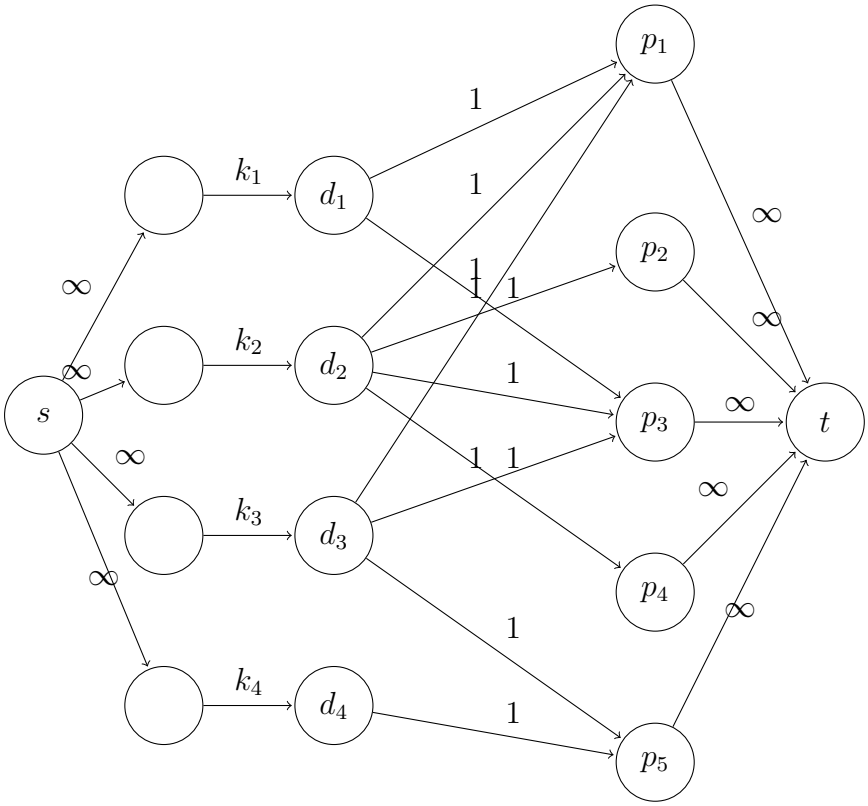
   Then, tack on two more vertices, s, t as the source and sink. Construct an edge from s to each $d_i \in L$ with weight ∞. Also construct an edge from each $p_j \in R$ to t with weight ∞.

   This is exactly the construction for the maximum bipartite matching problem. However, we have an additional constraint. Each doctor can treat at most $k_i > 0$ patients.

   We can modify our graph to encode this information by taking each edge $(s, d_i) \forall d_i \in L$ and creating an additional vertex and edge between them. Give this new edge a weight of k_i.

   Now, we can run FORD-FULKERSON over our newly constructed graph, and we shall have a maximum flow which represents the most number of treated patients.

   An example is shown below:



3. (a) True.

$$v(f) = f_{OUT}(s) - f_{IN}(s) = f_{OUT}(s) = f_{IN}(u_1) = f_{OUT}(u_1) = f_{IN}(u_2) = \cdots = f_{IN}(t)$$

(b) False. Minimum cuts are equivalent to maximum flow by the Max-flow min-cut theorem.

(c) False. The while loop in FORD-FULKERSON has complexity $O(|E|)$ since the path can be found with either BFS or DFS, both of which have complexity $O(|E|)$.

(d) True. This satisfies both the capacity and the conservation constraints.