CS345: Assignment 3

Pragati Agrawal (220779)

October 2024

Pragati Agrawal: 220779 (Question 1)

Description of the Algorithm:

This problem can be solved by reducing it to an instance of the Max-Flow problem. We can view this problem as a directed graph $G = (V, E)$, such that

- V is the set of size $(n + k)$, consisting of all buses B and stations S.
- E is the set of directed edges, from buses to stations, such that all stations reachable (i.e. within distance r) from any bus have an edge from that bus to that station. (There are no edges between two bus vertices or two station vertices).

Now, we add a source and sink to the graph to create a graph $G' = (V', E')$, such that there is no incoming edge on the source vertex s and no outgoing edge from the sink vertex t. There are edges from s to all buses, and also from all stations to the sink t[1].
Now we assign capacities $c : E' \to R^+$ to all the edges as follows:

- For any edge (s, u), such that s is the source vertex and $u \in B$, we assign its capacity $= 1$.
- For any edge (u, v), such that $u \in B$ and $v \in S$ reachable from u, we assign its capacity $= 1$.
- For any edge (v, t), such that $v \in S$ and t is the sink vertex, we assign its capacity $= L$, the capacity constraint.

Our aim is to compute a *matching* such that each of the n buses is connected to exactly one reachable station, and also the capacity constraint is satisfied for each station. By *matching* we mean a subset $P \subseteq E'$, such that all edges in P are of the from (u, v), where $u \in B$ and $v \in S$. In a *matching*, $\forall u \in B$ there can be atmost **1** outgoing edge from u in P and $\forall v \in S$, there can be atmost **L** incoming edges to v in P, such that the total number of outgoing edges from the bus vertices combined, is equal to the total number of incoming edges to the station vertices combined.
We will show that for any flow of value x in G', there will be a *matching* of size x in the original problem and vice-versa (i.e. there will be x distinct buses which will be matched to a reachable station, such that the capacity parameter is satisfied for each station).
So if we compute the maximum flow from s to t in G', we will get the maximum *matching* of the problem, because we cannot get any *matching* of size greater than the maximum flow value. Now if the maximum *matching* is of size less than n, we can say that every bus cannot be connected simultaneously to a station, subject to the range and load conditions. Trivially, the maximum possible *matching* size is n.

[1]Throughout the solution, assume s is the source, u is a bus vertex, v is a station vertex, and t is the sink.

Proof of correctness for the Algorithm:

We will show that there is a matching of size x (i.e. x buses can be matched to some station), iff there is a flow of value x in the corresponding graph G'.
There are two directions for the proof:

For any *matching* P of size x, we will have a flow of value x in G'.
Proof: Consider any valid *matching* P of size x. We assign a flow $f(u,v) = 1$ to all the edges in $(u,v) \in P$ and $\forall u \in B$, such that $(u,v) \in P$, we assign flow $f(s,u) = 1$. Similarly, $\forall v \in S$, we assign flow $f(v,t) = \textit{number of incoming edges to } v$.
For all remaining edges in E', we assign flow $f = 0$.

- *Capacity constraint:* For all edges of the type (s,u) and (u,v), we had assigned capacity = 1, and in the flow constructed above, we have flow on them = 1 or 0, so flow $\leq$ capacity. Now, for edges (v,t), we have assigned capacity L, and in the flow constructed above, they have flow = number of incoming edges. Since we have a valid *matching* P, we know that each station can have atmost L buses connected to it, so this number of incoming edges to any station vertex also has to be less than or equal to L. So flow $\leq$ capacity for all edges in G', thus capacity constraint is satisfied.

- *Conservation constraint:* We will show this for all intermediate vertices, i.e. $\forall u$ and $\forall v$:
For bus vertices u, if we have an edge $(u,v) \in P$, then we know that $f(s,u) = 1$ and $f(u,v) = 1$ by construction of the flow. Note that since P is a valid *matching*, there cannot be any edge from this u to any other vertex v carrying a positive flow, i.e. for all other station vertices $v' \in S$, s.t. $(u,v') \in E'$, we will have $f(u,v') = 0$. If for any u, no edge (u,v) is in P, then again by construction, $f(s,u) = 0$ and $f(u,v) = 0$, for all $v \in S$, s.t. $(u,v) \in E'$. Thus conservation is satisfied at all bus vertices.
For station vertices v, we know that by construction of the flow, incoming flow at v = (number of incoming edges to v) $* 1 = f(v,t)$ = outgoing flow from v, since each incoming edge has a flow 1. Thus conservation is satisfied at all station vertices.

- *Value(f) = x:*
Consider the $s - t$ cut such that $A = \{s\} \cup B$ and $A' = \{t\} \cup S$. We know that $value(f) = f_{out}(A) - f_{in}(A)$. Since there is no incoming edge from A' to A, $f_{in}(A) = 0 \Rightarrow value(f) = f_{out}(A)$. Since each edge going out of A goes from $u \in B$ to $v \in S$ and has capacity = 1, it can have flow = 0 or 1. Since we have a *matching* of size x, there must be exactly x edges from B to S, each having flow = 1. Therefore, we have $f_{out}(A) = x * 1 = x = value(f)$.

For any flow of value x in G', we will have a *matching* P of size x.
Proof: Consider any valid flow of value x. By integrality theorem, we will always have a flow f of value x having integer flows at all edges in G'. By capacity constraint and integrality of f, we know that all $(u,v) \in E$ will have a flow of either 0 or 1. We pick the edges having $f(u,v) = 1$ and leave the remaining ones ($u \in B$ and $v \in S$). This set of edges is our *matching* P. We will prove that it is a valid *matching*, and its size is x.

- *For any bus vertex $u \in B$, there is atmost 1 outgoing edge in P.*
Since every u has only one incoming edge from s, $f_{in}(u) = f(s,u)$. By capacity constraint of any edge (s,u), we know that $f(s,u) \leq c(s,u) = 1$. Now, by conservation constraint, $f_{in}(u) = f_{out}(u) \Rightarrow f_{out}(u) \leq 1$. Since f is integral, each edge (u,v) can have flow = 0 or 1. Therefore, $f_{out}(u)$ is the number of edges going out of u, each having flow =1. Since $f_{out}(u) \leq 1$, there can be atmost one edge from u in P.

- *For any station vertex v, there are atmost L incoming edges in P.*
  Since every v has only one outgoing edge to t, $f_{out}(v) = f(v,t)$. By capacity constraint of any edge (v,t), we know that $f(v,t) \leq c(v,t) = L$. Now, by conservation constraint, $f_{in}(v) = f_{out}(v) \Rightarrow f_{in}(v) \leq L$. Since f is integral, each edge (u,v) can have flow = 0 or 1. Therefore, $f_{in}(u)$ is the number of edges coming into v, each having flow =1. Since $f_{in}(v) \leq L$, there can be atmost L edges to v in P.

- *There are exactly x edges in P.*
  Consider the $s-t$ cut such that $A = \{s\} \cup B$ and $A' = \{t\} \cup S$. Since $flow = x$, therefore, $value(f) = f_{out}(A) - f_{in}(A) = x$. Since there is no incoming edge from A' to A, $f_{in}(A) = 0 \Rightarrow f_{out}(A) = x$. Since each edge going out of A has capacity = 1, it can have flow = 0 or 1. Since $f_{out}(A) = x$, there must be exactly x edges having flow = 1, going out of A. We know that all edges going from A to A' go from $u \in B$ to $v \in S$, and we have chosen all such edges in P. Therefore, we will have exactly x edges going from B to S in P, thus $|P| = x$.

Hence we have proved that there is a *matching* of size x iff there is a flow of value x in the corresponding graph G'. Therefore, we can say for any flow, we will get a corresponding *matching* of the same size, and for every *matching*, we will get a flow of the same value. So if the Max-Flow has value x_{max}, there cannot be any flow of value greater than x_{max}. By the theorem, we will get a *matching* of size x_{max}, and we cannot have any *matching* of size greater than x_{max}. So x_{max} is the maximum number of buses that can be connected to a station, satisfying the constraints. Hence our algorithm's correctness is proved.

Pseudo-code:
Assume **B** is the set of bus vertices, **S** is the set of station vertices, and r and L are the input parameters. The function *ComputeMaxFlow* computes the Max-Flow in a given graph.

Algorithm 1 FindSolution (B, S, r, L)

```
1: FindSolution (B, S, r, L){
2:     $V' \leftarrow \{s\} \cup \{t\} \cup B \cup S$
3:     $E' \leftarrow (s,u) \cup (v,t)$
4:     $E' \leftarrow E' \cup (u,v)$ ($\forall$ $u \in B$ and $v \in S$, such that $(u,v) \in E$ i.e. $dist(u,v) \leq r$)
5:     assign $c(s,u) = 1$, $c(u,v) = 1$, $c(v,t) = L$
6:     let $G' = (V', E')$
7:     x $\leftarrow ComputeMaxFlow(G')$
8:     if $(x == n)$ return true
9:     else return false
10: }
```

Time Complexity Analysis: The algorithm would run in $O(M^2N)$ using Edmond-Karp method, where number of edges $M = O(n + n * k + k) = O(n * k)$ and number of vertices $N = O(n + k + 2)$, i.e. $O(n^2k^2(n + k))$, which is a polynomial time algorithm.

Pragati Agrawal: 220779 (Question 2)

Description of the Algorithm:

We would modify the given graph such that this problem is reduced to an instance of the Edge-Disjoint paths problem in a directed graph. For this, we would add a duplicate of every vertex for each vertex in the graph, say x' for each vertex x, and add an edge $x \to x'$. All edges incoming on x would be intact, but all edges outgoing from x would be removed from x, and made as outgoing edges of x'.
Let the original graph be $G = (V, E)$ and the new graph we have constructed be $G' = (V', E')$. We will show that the number of edge-disjoint paths in G' from s' to t will be the same as the number of vertex-disjoint paths in G from s to t.

Proof of correctness for the Algorithm:

We will show that the number of edge-disjoint paths in G' from s' to t is equal to the number of vertex-disjoint paths in G from s to t. There are two parts to the proof:

There is a one-to-one mapping between an $s \to t$ path P in G and a corresponding $s \to s' \to ..... \to t \to t'$ path P' in G'.
Proof: We will give proof by construction. Consider any $s \to t$ path P in G. Now for any edge $(u, v) \in P$, we will take the edges $u \to u' \to v \to v'$ in G' as a part of P', and keep doing this until we reach t' (such a path from $u \to v'$ will always exist by construction of G'). This gives us the path P' corresponding to P. Note that $|P'| = 2 * |P|$.
Now, for uniqueness, consider two distinct paths P_1 and P_2 that yield the same P'. If their lengths are different, then we cannot have the same P', as the length of P' would be different for both. Now, if their lengths are the same, they must differ in atleast one original vertex. But then by the construction of P', atleast that one original and its duplicate vertex must be different in P' for both the paths. This is not possible, since we assumed that both yield the same P'. Hence there will be a unique path P corresponding to every P'.

Similarly, for the reverse side, consider any $s \to s' \to ..... \to t \to t'$ path P' in G'. To construct an $s \to t$ path P in G, we choose all the vertices at odd positions in P' until we reach t (such a path will always exist in G, by construction of G'). This chain of vertices constitutes the path from s to t in G. Note that $|P| = |P'|/2$.
Now, for uniqueness, consider two distinct paths P'_1 and P'_2 that yield the same P. If their lengths are different, then we cannot have the same P, as the length of P would be different for both. Now, if their lengths are the same, they must differ in atleast one duplicate vertex, (because if P is the same for both, all the odd position original vertices must be the same in P'_1 and P'_2, by construction of P). Again, if all odd position vertices are same in P'_1 and P'_2, then we know that by construction of G', we have only one outgoing edge from any original vertex in G' which goes to its duplicate vertex. So these must be the same in both P'_1 and P'_2. Hence there will be a unique path P' corresponding to every P.

If any two $s \to t$ paths P_1 and P_2 in G are vertex-disjoint (excluding the vertices s and t), then their corresponding $s \to s' \to ..... \to t \to t'$ paths P'_1 and P'_2 in G' would be edge-disjoint (excluding the edges $s \to s'$ and $t \to t'$), and vice-versa.

Proof: We give proof by contradiction.

- P_1 and P_2 are vertex-disjoint $\Rightarrow$ P'_1 and P'_2 are edge-disjoint.
  Assume P'_1 and P'_2 are not edge-disjoint, i.e. they share atleast one edge. This can be of the form $x \to x'$ or $x' \to y$. If they share an edge $x \to x'$, this means that both the paths P_1 and P_2 share the vertex x, by construction of P from P'. This is not possible since P_1 and P_2 are vertex-disjoint. If they share an edge $x' \to y$, then they must also share the edge $x \to x'$, since there is only one incoming edge to x'. This reduces us to the first case, and again this is not possible.

- P'_1 and P'_2 are edge-disjoint $\Rightarrow$ P_1 and P_2 are vertex-disjoint.
  Assume P_1 and P_2 are not vertex-disjoint, i.e. they share atleast one vertex. Let that vertex be x. Then by construction of P' from P, we know that we will have edge $x \to x'$ in both P'_1 and P'_2. This makes P'_1 and P'_2 not edge-disjoint, but this is not possible. Therefore, P_1 and P_2 must be vertex-disjoint.

Using the above claims, we can say that for every vertex-disjoint path in G from s to t, we will get a unique edge-disjoint path in G' from s' to t and for every edge-disjoint path in G' from s' to t, we will get a unique vertex-disjoint path in G from s to t. We need to ignore the edges $(s \to s')$ and $t \to t'$ while applying the Edge-Disjoint paths problem, because these edges would be common to all paths (since vertices s and t are common to all vertex disjoint paths). So we apply the algorithm from s' to t.

Pseudo-code for the Algorithm:

Assume **G** is the given input graph, s is the source vertex, and t is the sink vertex. The function *FindEdgeDisjoint* computes the maximum number of edge-disjoint paths in a given graph between s and t.

Algorithm 2 FindVertexDisjoint (G, s, t)

```
1: FindVertexDisjoint $(G, s, t)${
2:     $V' \leftarrow \emptyset,\ E' \leftarrow \emptyset$
3:     $V' \leftarrow V' \cup \{x, x'\}$ (for every vertex $x \in V$)
4:     $E' \leftarrow E' \cup (x, x')$ (for every vertex $x \in V$)
5:     $E' \leftarrow E' \cup (x', y)$ (for every edge $(x, y) \in E$)
6:     let $G' = (V', E')$
7:     k $\leftarrow$ FindEdgeDisjoint$(G', s', t)$
8:     return $k$
9: }
```

Time Complexity Analysis:

The edge-disjoint paths algorithm is essentially an application of the max-flow algorithm, so it works in $O(m^2 n)$ time, using the Edmond-Karp method.