Maximum Flow Problem

Oliver-Matis Lill

June 16, 2017

Mathematical Notation

- In our slides we use the following mathematical symbols from outside the high school curriculum:

1. $\forall x \in S : P(x)$ - For every x from the set S the condition $P(x)$ is true
2. $\sum_{x \in S} g(x)$ - The sum of $g(x)$ over all x from the set S. Basically equivalent to $g(x_1) + g(x_2) + \ldots + g(x_n)$ if $S = \{x_1, x_2, \ldots, x_n\}$
3. $S \setminus T$ - A subset of elements from S that are not in T. Basically set subtraction

The Maximum Flow Problem

- You have a directed graph with a set of edges E and a set of vertices V
- Each edge e has some non-negative capacity c_e
- The set of edges leaving a vertex v is out(v) and the set of edges entering it is: in(v)

The Maximum Flow Problem

- Given the source vertex s and sink vertex t assign a flow f_e to each edge e such that the following conditions are satisfied:
  1. $\forall v \in V \setminus \{s, t\} : d(v) = 0$
  2. $\forall e \in E : 0 \leq f_e \leq c_e$

  Where $d(v)$ is the flow differential of a vertex v, defined as:

$$d(v) = \sum_{e \in \text{out}(v)} f_e - \sum_{e \in \text{in}(v)} f_e$$

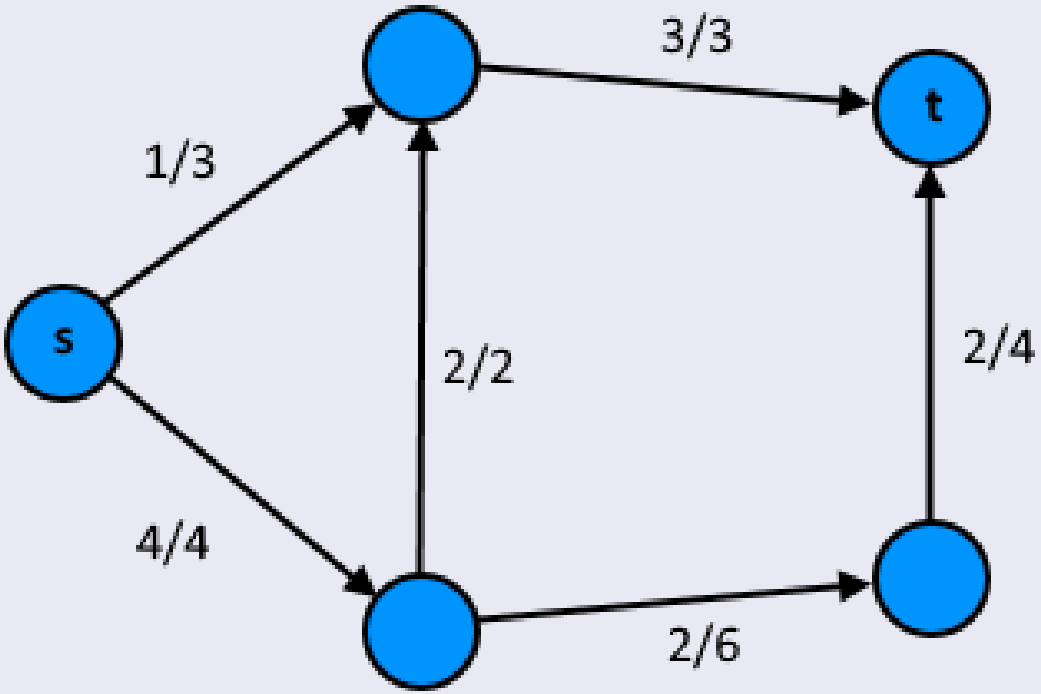
- Our goal in the assignment is to maximize the network flow:

$$d(s)$$

- For simplicity suppose that each f_e must be an integer

Maximum Flow Example

Example



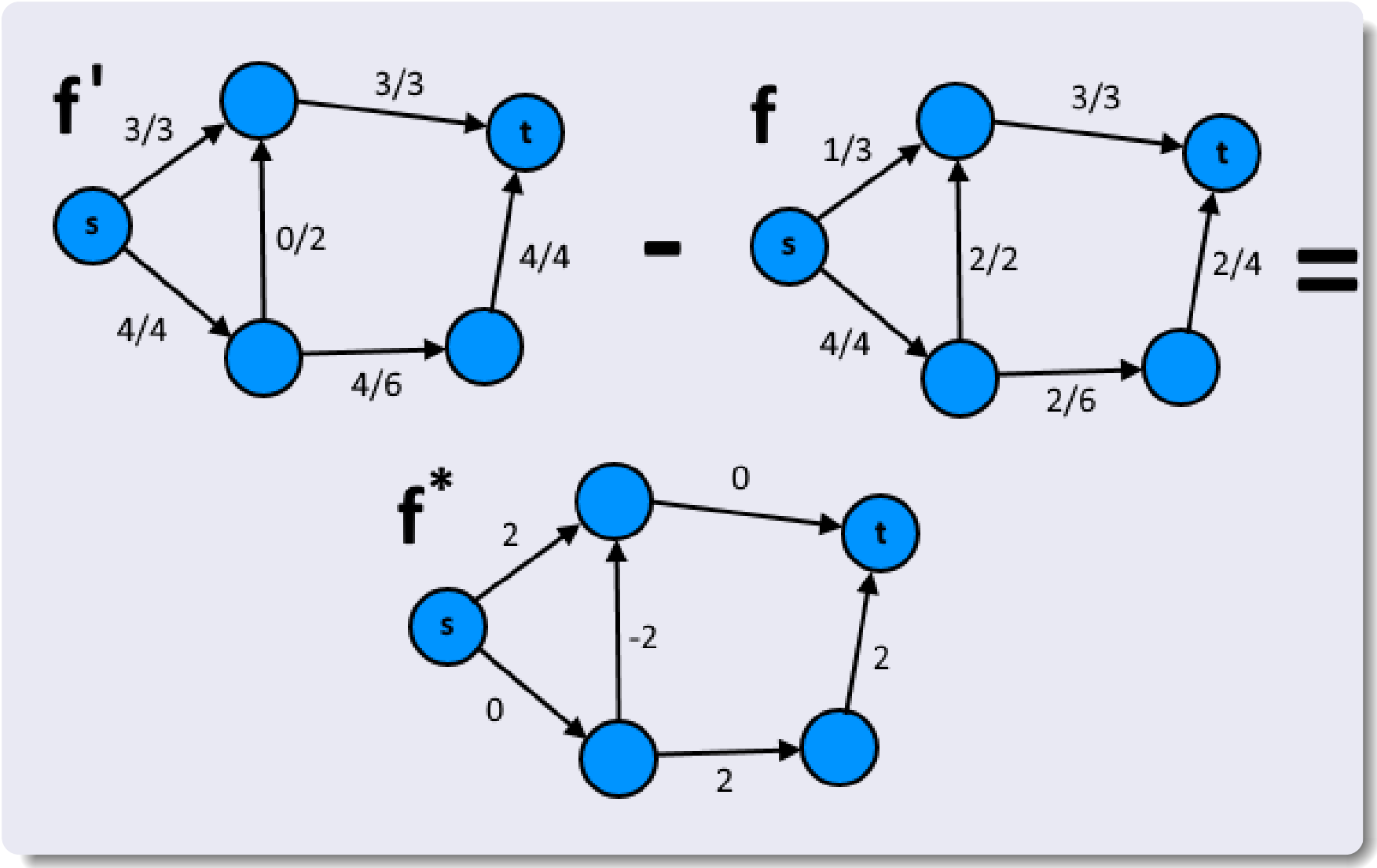
Maximum Flow Attributes

- Prove that that:
$$d(s) = -d(t)$$
- How do we know if some flow assignment is optimal?
- If you have heard of augmenting paths, why does one always exist for an unoptimal assignment?
- We will be focusing on a proof to the latter

Maximum Flow Idea

- Suppose for each edge our current flow is f_e but in some optimal solution it would be f'_e
- Let's denote $f^*_e = f'_e - f_e$
- Let's denote that $d^*(v) = \sum_{e \in \text{out}(v)} f^*_e - \sum_{e \in \text{in}(v)} f^*_e$
- Note that here too $\forall v \in V \setminus \{s, t\} : d^*(v) = 0$

Maximum Flow Example



Maximum Flow Idea

- Note that in case of unoptimal solution, there always exists a path from s to t by traversing forward on edges where $f_e^* > 0$ and backwards where $f_e^* < 0$, thanks to the fact that $d^*(v) = 0$ if v is not s or t
- By increasing flow by 1 on edges traversed forward and decreasing by 1 on edges traversed backwards, we'll get a valid flow network with greater $d(s)$
- This is called an "augmenting path" and as we have shown, it always exists when flow assignment is not optimal

Maximum Flow Code

```
bool augment(Node* v) {
    if(v == t) return true;
    if(v->explr) return false;
    v->explr = true;
    //Traverse forward
    for(Edge* e : out(v))
        if(e->flow < e->cap && augment(e->end)) {
            e->flow += 1;
            return true;
        }
    //Traverse backwards
    for(Edge* e : in(v))
        if(e->flow > 0 && augment(e->start)) {
            e->flow -= 1;
            return true;
        }
    return false;
}
```

Maximum Flow Conclusion

- By running the aforementioned algorithm repeatedly, we eventually reach an optimal flow assignment
- What is it's time complexity?
- This is called the Ford-Fulkerson algorithm
- This approach of looking at some theoretical optimal solution and subtracting the current one is quite useful. It can be used to prove various Min-Cost Max-Flow algorithms