Exercise 26.1-1

Show that a maximum flow in G' has the same value as a maximum flow in G

Let G, G', x, u and v be given as in the problem description.

What we want to argue here is that any flow in G is also feasible in G' and with the same flow value. Then we conclude that $|f^*_G| = |f^*_{G'}|$ where $|f^*_G|$ is the maximum flow value in the network G.

First observe that any flow $f(u,v)$ in G can be replaced with a equivalent flow in G' where $f(u,v) = f(u,x) = f(x,v)$ since $c(u,v) = c(u,x) = c(x,v)$.

Let $p_{f^*_G}$ be the path the maximum flow f^* takes in G.

If $(u,v) \notin p_{f^*_G}$

In this case the same path is available in G' which would yield the same flow value. If introducing the new vertex x means that $p_{f^*_{G'}}$ is changed such that $\{(u,x),(x,v)\} \subset p_{f^*_{G'}}$, then by our first observation above, the same flow value is possible in G, and since $(u,v) \notin p_{f^*_G}$ deviating cannot yield a higher flow value.

If $(u,v) \in p_{f^*_G}$

In this case we have already observed that replacing the $f(u,v)$ with $f(u,x) = f(x,v)$ will yield the same flow value. Hence the maximum flow value will be the same.

If introducing the new vertex x in G' changes the flow value from u to v, then this cannot yield a higher maximum flow value in G'. This is because of our observation above, as all flow from u to v are bounded by the same capacity constraint in both G and G', the same flow value is feasible in both networks. As f^*_G is part of a maximum flow value, deviating from that cannot yield a higher flow value, as that goes against it being a maximum flow value in the beginning.

Hence we conclude that in all possible cases the maximum flow value will be the same, hence $|f^*_G| = |f^*_{G'}|$.

Exercise 26.1-4

Prove that the flows in a network form a convex set.

Let f_1, f_2, α be as given in the exercise description.

We want to prove that $f_\alpha \equiv \alpha f_1 + (1-\alpha) f_2 \ \forall \alpha \in (0,1)$ is a flow.

We start by showing the capacity constraint i.e.

$$\forall u, v \in V \text{ it holds that } 0 \leq f_\alpha(u, v) \leq c(u, v). \tag{1}$$

Since f_1 and f_2 are flows, the capacity constraint holds for both of them. Since the capacity function c is a property of the network, and not the individual flows, the capacity constraint for f_1 and f_2 are identical to (1) i.e they are bounded by the same upper and lower bounds.

Hence $\forall \alpha \in (0, 1)$ and $\forall u, v \in V$ the following equations hold

$$\begin{aligned} f_\alpha(u, v) &= \alpha f_1(u, v) + (1-\alpha) f_2(u, v) \\ &\geq \alpha 0 + (1-\alpha) 0 \\ &\geq 0 \end{aligned}$$

and

$$\begin{aligned} f_\alpha(u, v) &= \alpha f_1(u, v) + (1-\alpha) f_2(u, v) \\ &\leq \alpha c(u, v) + (1-\alpha) c(u, v) \\ &= c(u, v) \end{aligned}$$

Combining the two inequalities above yields that (1) holds.

We then want to show that the flow conservation property holds. i.e

$$\forall u \in V \setminus \{s, t\} \text{ it holds that } \sum_{v \in V} f_\alpha(v, u) = \sum_{v \in V} f_\alpha(u, v). \tag{2}$$

First note that since f_1 and f_2 are flows, the flow conservation property holds for both of them individually.

Pick a vertex u such that $u \in V \setminus \{s, t\}$ then

$$\begin{aligned} \sum_{v \in V} f_\alpha(v, u) &= \sum_{v \in V} (\alpha f_1(v, u) + (1-\alpha) f_2(v, u)) \\ &= \alpha \sum_{v \in V} f_1(v, u) + (1-\alpha) \sum_{v \in V} f_2(v, u) \\ &= \alpha \sum_{v \in V} f_1(u, v) + (1-\alpha) \sum_{v \in V} f_2(u, v) \\ &= \sum_{v \in V} (\alpha f_1(u, v) + (1-\alpha) f_2(u, v)) \\ &= \sum_{v \in V} f_\alpha(u, v) \end{aligned}$$

Since this holds for all $u \in V \setminus \{s, t\}$, we have now shown (2).

We have now shown the capacity constraint and the flow conservation property for f_α, hence f_α is a flow. So the flows in a network form a convex set.

Exercise 26.1-7

Show that a flow network G with vertex capacities can be transformed into a normal flow network G'. How many vertices and edges does G' have?

Let the G, G', v and l be defined as in the exercise.

First we describes how a network G_x where only one of the vertices, x, have a capacity constraint $l(x)$ and how it can be transformed into a new network without the vertex capacity constraint.

We define the set $E(x)$ of edges that have an edge going into the vertex with a capacity constraint. That is $E(x) = \{(v, x) \in E : l(x) > 0\}$.

To transform the network, we can augment G_x with a new vertex x' such that $G'_x = (V', E')$ where $V' = V \cup \{x'\}$. We define E' as $E' = E \setminus E(x) \cup E'(x) \cup \{(x', x)\}'$ where $E'(x)$ is the set of edges in $E(x)$, but with entrance point into the new vertex x'. So the set of edges E' in the new transformed network consist of all edges in the original network, that did not have an edge going into the vertex with a capacity constraint. And all the edges that did go into the vertex with a capacity constraint have been replaced with new edges that goes into the new vertex x'. And a new edge have been introduced that goes from the new vertex x' to the old vertex x. Hence in total one new vertex and one new edge have been introduced and none have been removed.

Figure 1: *Augmenting the network G_x to G'_x*

(a) Introducing the new vertex x'

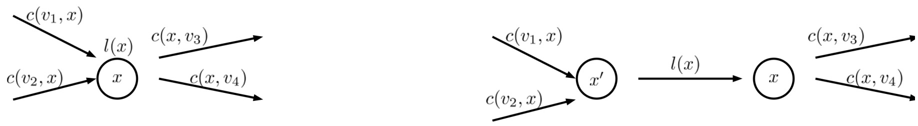
The capacity of the edges in the transformed network, where no vertex have a capacity is

$$c_{G'_x}(u, v) = \begin{cases} c_{G_x}(u, x), & (u, v) = (u, x') \\ l(x), & (u, v) = (x', x) \\ c_{G_x}(u, v), & \text{else} \end{cases}$$

That is, the capacity for an edge going into x' is the same as the capacity of going into x in the G_x network. The new edge have a capacity of $l(x)$ that is the capacity of the vertex in the G_x network. In all other cases the capacity in the transformed network is the same as in the original network.

Figure 2: *Augmenting the network G_x to G'_x*

(a) Setting the edge capacities



We have now shown how a network with a single vertex capacity constraint can be transformed into a new network without a capacity constraint. This can easily be generalized to an arbitrary number of vertex capacity constraint. If x and x' is replaced with x_i and x'_i in the above, all of what we have shown still holds, and for each vertex with a capacity constraint, there will be one new vertex and one new edge. Hence if all of the vertices, including the source and sink, had a constraint there would be $|V|$ extra vertices and edges. So the total number of vertices in G' would be $2|V|$ and the total number of edges in G' would be $|E| + |V|$.

Exercise 26.2-2

What is the flow and cut

Let the cut be given as in the exercise, then the flow across the cut (and all other cuts) in figure 26.1(b) is

$$\begin{aligned} f(S,T) = f(\{s, v_2, v_4\}, \{v_1, v_3, t\}) &= \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u) \\ &= 11 + 1 + 7 + 4 - 4 = 19 \end{aligned}$$

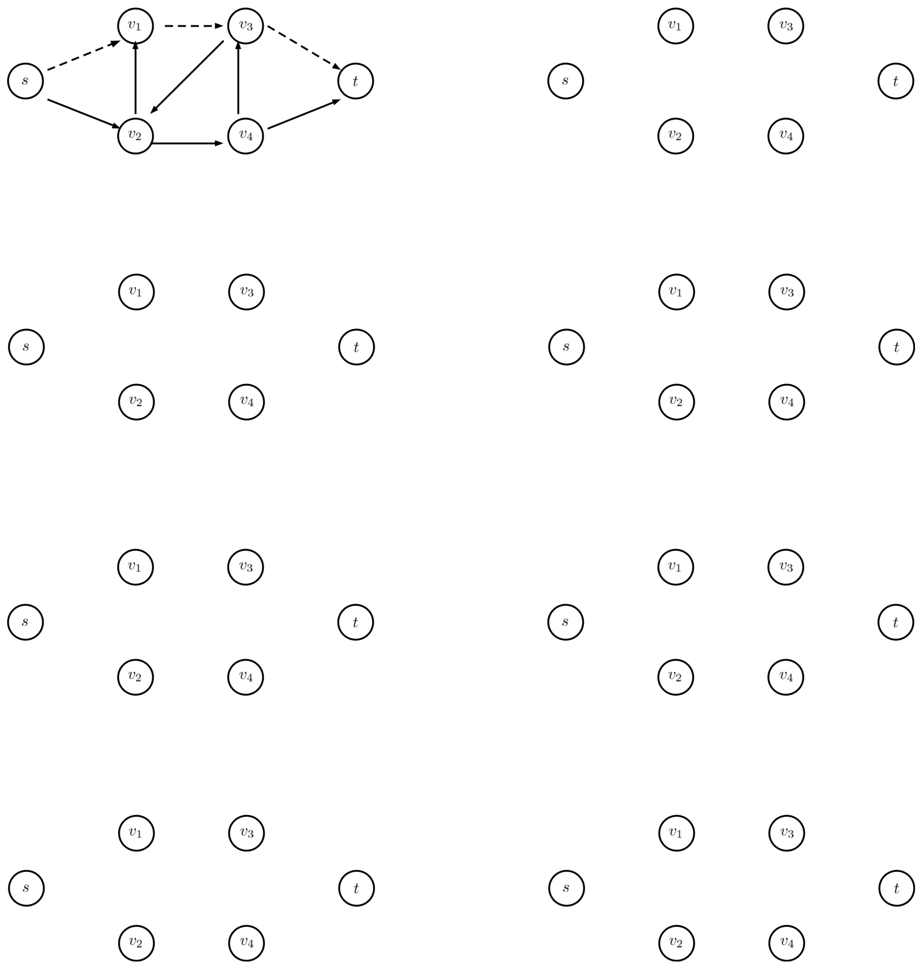
The capacity is

$$C(S,T) = \sum_{u\in S}\sum_{v\in T} c(u,v) = 16 + 4 + 7 + 4 = 31$$

1 Exercise 26.2-3

Figure 3: *Executing the Edmonds-Karp algorithm.*

(a) Blablablablablabla



2 Exercise 26.2-4

In the example of Figure 26.6, the maximum flow shown has the value 23. The minimum cut corresponding to this is

$$(\{s, v_1, v_2, v_4\}, \{v_3, t\}),$$

since the capacity is

$$c(v_1, v_3) + c(v_4, v_3) + c(v_4, t) = 12 + 7 + 4 = 23.$$

The augmenting path in (c) cancels flow:
4 units on (v_1, v_2) and (v_2, v_3).

3 Exercise 26.2-7

Proof of Lemma 26.2.

We first prove, that f_p is a flow in G_f, by proving the capacity constraint and flow conservation property holds in G_f.

Capacity constraint:
It is given, that $f_p(u, v) = c_f(p)$, if (u, v) is on the path p. We also know that the residual capacity $c_f(p)$ is the minimum capacity of any edge on p, which gives us the upper bound on the flow through all edges on the path
$f_p(u, v) \leq c_f(u, v)$
Furthermore, it is also given, that $f_p(u, v) = 0$, if (u, v) is not on p, which gives the lower bound on the flow
$0 \leq f_p(u, v)$.

Flow conservation property:
As noted earlier, the flow on p will be determined by the residual capacity $c_f(p)$ of p, meaning the same flow will be pushed through all edges on p. Hence, the flow conservation property holds.

Proof that $|f_p| = c_f(p) > 0$:
We know from the definition of the value of a flow, that

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

The source s must have at least one more outgoing edge than ingoing. Due to the flow conservation property, flow on ingoing and outgoing edges must even out, so we only need to concern ourselves with the "extra" outgoing edge. In this case, there is no ingoing flow for s, so

$$|f_p| = \sum_{v \in V} f_p(s, v) - \sum_{v \in V} f_p(v, s) = c_f(p) - 0 = c_f(p)$$

4 Exercise 26.2-9

We are given to flows, f and f' and compute the augmented flow $f \uparrow f'$.

Does the augmented flow satisfy the flow conservation property?
Yes!

For any vertice $u \in V - \{s, t\}$, one of the following cases apply:
Neither f nor f' includes u.
There is no flow, so $f(u, v) = f(v, u)$, where $v \in V$.

f and/or f' includes u.
It is given, that f and f' are ordinary flows, so the flow conservation property must hold for each of these; all flow into a vertice will also flow out. Regardless of whether u is included in one or two flows, all flow into u will also come out.

Does the augmented flow satisfy the capacity constraint?
No!

Consider the following case:
The edge (u, v) has capacity $c(u, v) = 2$. The edge (u, v) is part of both flows f and f', where $f(u, v) = 2$ and $f'(u, v) = 2$.
Hence, $f(u, v) + f'(u, v) = 2 + 2 = 4 > c(u, v)$.

5 Exercise 26.3-2

Proof of Theorem 26.10.

First, the Ford-Fulkerson method initializes the flow to 0. Then, in every iteration, the flow is augmented with the residual capacity, $c_f(p)$, of the augmenting path. Since $c_f(p)$ is set from the integer residual capacity of the edges, $|f|$ will always sum up to an integer.

Consider iteration 0 (no augmenting path):
$|f|$ is initiated to 0, which is an integer.
Consider iteration $n + 1$:
If $|f|$ was an integer after the nth iteration, then in the $n + 1$th iteration, we will augmenth $|f|$ with an integer, which in turn results in an integer.