Imperial College London

Department of Computing

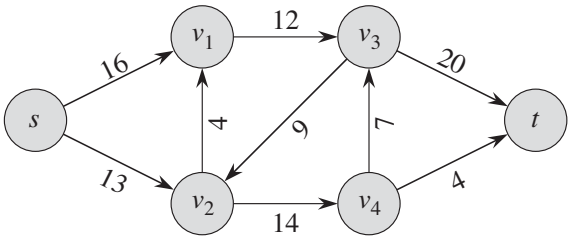
CO-202 Algorithms II, Autumn 2018

Tutorial sheet 8

Exercise 1

(Max-flow/Min-cut)

1. Find the maximum flow from s to t, along with the associated minimum cut, in the network depicted below. Show the residual network and augmenting path at each step of the Ford-Fulkerson algorithm.



2. Using the Max-Flow/Min-Cut theorem, show that if G is a flow network where all capacities are integers, then the maximum flow is also integer-valued.

3. Let G be a flow network for which all capacities are even numbers. Using the Max-Flow/Min-Cut theorem, show that the maximum flow is also an even number.

4. Suppose you have a flow network G. While walking in a sketchy part of town, a stranger offers you a solution f to the max-flow problem on G. Of course, you do not trust strangers (especially those who solve random max-flow problems in a back alley), so you are not sure whether f is a solution to your problem. Design an algorithm that determines whether f is indeed a maximum flow in time $O(|V|+|E|)$.

Model Answer

1. This exercise is given as a solved example in *Introduction to Algorithms* by Cormen et al., Section 26.2. We showcase the steps of the Ford-Fulkerson algorithm in Figure 1. The graphs on the left hand side represent the successive residual networks. The augmenting path to be added to the flow is highlighted. The graphs on the

right hand side represent the flows obtained in the intermediate steps of the Ford-Fulkerson algorithm. Note that there is no fixed way of choosing the augmenting path for each residual network. Therefore, one may obtain different intermediate networks, and even a different maximum flow function. The maximum flow of this network is $|f| = 23$.

To find a minimum cut in the network, it suffices to find a cut (S, T) such that $s \in S$ and $t \in T$ of size $c(S, T) = |f| = 23$. We can take $S = \{s, v_1, v_2, v_4\}$ and $T = \{v_3, t\}$. Then, we have

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v) = c(v_1, v_3) + c(v_4, v_3) + c(v_4, t) = 12 + 7 + 4 = 23.$$

Since the maximum flow of the network is 23, we conclude that (S, T) as defined above is a minimum cut.

2. According to the Max-Flow/Min-Cut theorem, the maximum flow of G equals the size of a minimum cut of G. Since all capacities $c(u, v)$ are integers for all $u, v \in V$ and the size of a cut (S, T) is given by

$$\sum_{u \in S, v \in T} c(u,v),$$

it follows that the maximum flow is also an integer.

3. Since all capacities are even numbers, the size of a minimum cut of G must also be even. By the Max-Flow/Min-Cut theorem, we conclude that the maximum flow of G is even.

4. By the Max-Flow/Min-Cut theorem, we know that a flow f is a maximum flow for G if and only if there is no augmenting path in the residual network G_f. We can check whether this property holds for the candidate solution f in time $O(|V| + |E|)$ with a BFS.

Exercise 2

(Applications of max-flow/min-cut)

The concepts of flow networks and maximum flow are fundamental in computer science because many seemingly unrelated algorithmic tasks can be formulated as maximum flow problems. In this exercise, your goal is to reduce a couple of graph-theoretic problems to the task of finding the maximum flow in an associated flow network. Besides describing the reduction itself, you must prove that a maximum flow in the associated flow network yields a solution to the original problem.

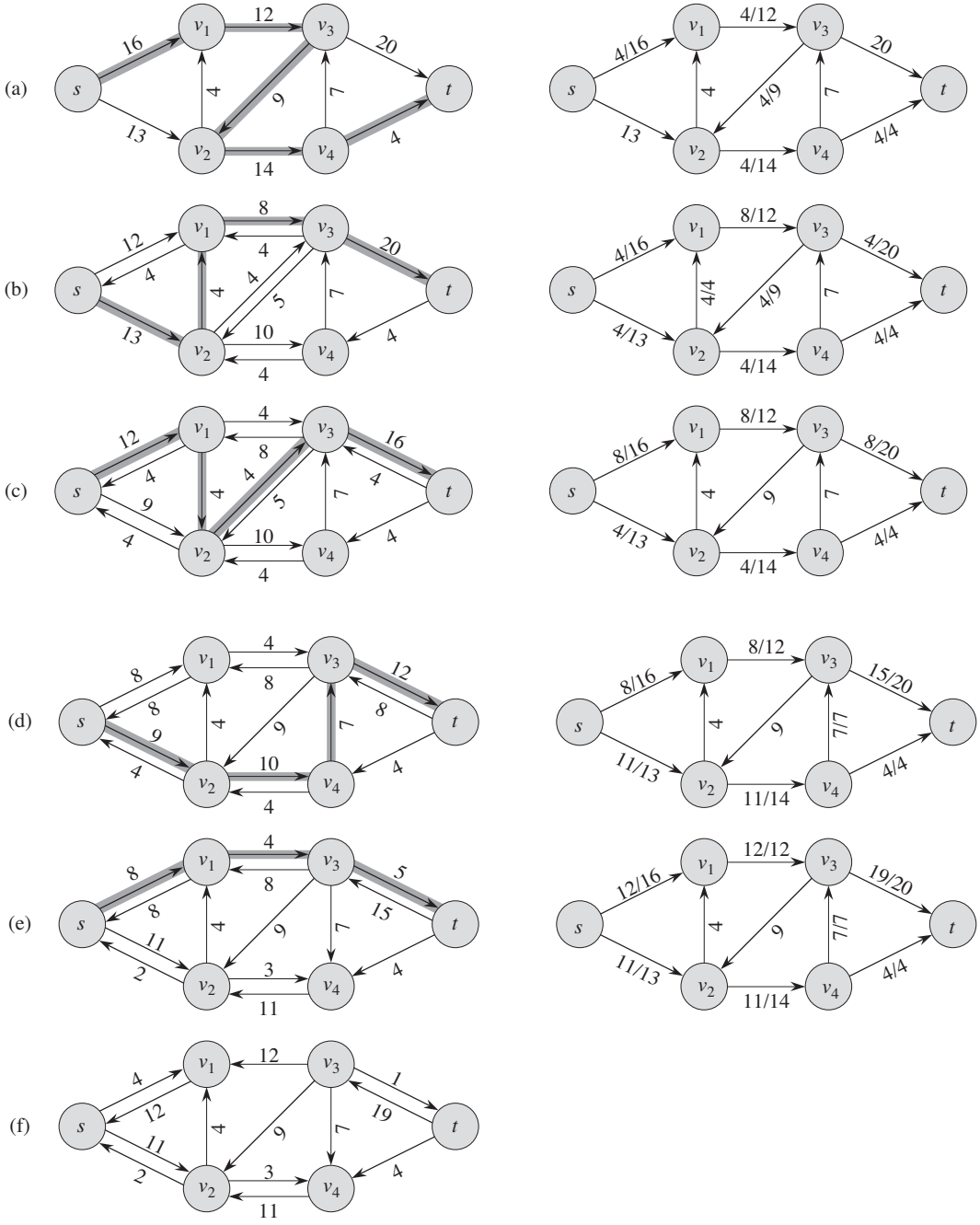


Figure 1: Residual networks and intermediate flows obtained during an execution of the Ford-Fulkerson algorithm. Note that there is no augmenting path in the last residual network. As a result, we know that our flow function is optimal.

Important remarks: A flow function f such that $f(e)$ is an integer for all edges e is said to be *integral*. It is easy to see that the Ford-Fulkerson method always returns an integral maximum flow whenever all the capacities in the network are integers. In particular, this means that there always exists an integral maximum flow in a network with integer capacities.

A graph $G = (V, E)$ is said to be *bipartite* if its vertex set V can be partitioned into two disjoint subsets L (the left vertex set) and R (the right vertex set) such that if $(u, v) \in E$, then either $u \in L$ and $v \in R$, or $u \in R$ and $v \in L$. We may also write $G = (L, R, E)$ for a bipartite graph.

1. Given an unweighted directed graph $G = (V, E)$, a set of paths $\{P_1, \ldots, P_k\}$ between a source s and a sink t is said to be *edge-disjoint* if each edge of G shows up in at most one path P_i. The goal is to compute the largest set of edge-disjoint paths in G.

   BONUS (non-examinable): Using the max-flow characterization of the edge-disjoint problem along with the Max-Flow/Min-Cut theorem, prove *Menger's theorem*:

   - The maximum number of edge-disjoint paths between s and t equals the minimum number of edges that must be removed from G to disconnect s and t.

2. Given an undirected bipartite graph $G = (L, R, E)$ with left vertex set L and right vertex set R, we wish to find the *maximum bipartite matching* of G. A bipartite matching of G is a set of edges $\{(\ell_1, r_1), \ldots, (\ell_k, r_k)\}$ with $\ell_i \in L$ and $r_i \in R$ such that no two edges share a vertex, i.e., it holds that $\ell_i \neq \ell_j$ and $r_i \neq r_j$ for all $i \neq j$. The maximum bipartite matching is the bipartite matching of largest size.

   BONUS (non-examinable): A graph is called k-regular if all its vertices have degree k. Let $G = (L, R, E)$ be an undirected k-regular bipartite graph with $|L| = |R|$. We call a bipartite matching of G *perfect* if all vertices are touched by some edge in the matching. In other words, for each vertex $u \in L \cup R$, there is an edge of the form (u, v) in the matching, for some $v \in V$. Using the max-flow characterization of the maximum bipartite matching problem, show that G admits a perfect bipartite matching.

Model Answer

1. Given an unweighted directed graph $G = (V, E)$ with a source s and a sink t, consider the flow network obtained by setting the capacity of $(u, v) \in E$ to be $c(u, v) = 1$. We claim that an integral maximum flow f between s and t in this flow network corresponds to the largest set of edge-disjoint paths between s and t in G.

   First, observe that if the paths $P_1, \ldots, P_k$ between s and t in G are edge-disjoint, then we can obtain a flow function f for the associated network with $|f| = k$ by sending one unit of flow through each path P_i, i.e., we set

   $$f(u, v) := |\{i : P_i \text{ contains edge } (u, v)\}|.$$

The edge-disjointness property of $P_1, \ldots, P_k$ ensures that $f(u, v) \in \{0, 1\}$, and hence the capacity constraints of the network are satisfied. Moreover, the path structure of $P_1, \ldots, P_k$ ensures flow conservation. That $|f| = k$ is a direct consequence of the fact that $P_1, \ldots, P_k$ use k different edges leaving s.

Second, an integral flow f with $|f| = k$ yields a collection of k edge-disjoint paths $\{P_1, \ldots, P_k\}$. The capacity constraints and the integrality of f ensure that $f(u, v) \in \{0, 1\}$ for all edges (u, v). The paths can be obtained by taking all edges (u, v) such that $f(u, v) = 1$. If the resulting paths $P_1, \ldots, P_k$ were not edge-disjoint, then there is some vertex v in the flow network such that $P_1, \ldots, P_k$ use more edges ending at v than edges starting at v. This implies that flow conservation does not hold for f, which is false since we assume f is a valid flow function.

The desired equivalence now follows immediately.

BONUS: It is easy to see that if it suffices to delete k edges to disconnect s and t, then the number of edge-disjoint paths is at most k. To see the other direction, we consider a maximum flow f in the associated flow network. Suppose that $|f| = k$. Then, the Max-Flow/Min-Cut theorem states that the minimum cut separating s and t has size k. The result now follows by noting that each set of deleted edges that disconnects s and t induces a cut in the flow network.

2. Fix $G = (L, R, E)$, and consider the flow network obtained by adding a source s connected by an edge of capacity 1 to each $\ell \in L$, and a sink t connected by an edge of capacity 1 to each $r \in R$. Furthermore, set $c(\ell, r) = 1$ for all $(\ell, r) \in E$, and $c(\ell, r) = 0$ otherwise. We claim that a maximum flow in this flow network corresponds to a maximum bipartite matching in G.

   First, given a bipartite matching $\{(\ell_1, r_1), \ldots, (\ell_k, r_k)\}$ in G, we can obtain a flow function f by setting $f(\ell_i, r_i) = 1$, $f(s, \ell_i) = 1$, and $f(r_i, t) = 1$ for $i = 1, \ldots, k$, and $f(u, v) = 0$ otherwise. Capacity constraints are satisfied by definition, and flow conservation is ensured by the fact that no two edges in the matching share a vertex. If there is some vertex r such that $f(\ell, r) = 1$ and $f(\ell', r) = 1$, then both (ℓ, r) and (ℓ', r) would be in the matching, which cannot happen. An analogous argument holds when exchanging the roles of ℓ and r. Hence, it holds that f is a valid flow function.

   Second, given an integral flow function f with $|f| = k$, we can obtain a bipartite matching of size k by selecting all edges $(\ell, r) \in E$ such that $f(\ell, r) = 1$. As before, flow conservation of f ensures that the resulting set of edges is a bipartite matching.

   BONUS: Fix a k-regular bipartite graph $G = (L, R, E)$ with $|L| = |R| = n$. We consider the flow network associated to G described above. Define the flow f by setting $f(e) = 1/k$ for every $e \in E$, and $f(s, \ell) = f(r, t) = 1$ for all $\ell \in L$ and $r \in R$. Observe that f satisfies the capacity constraints of the network since $k \geq 1$. Moreover, since G is k-regular, it follows that f satisfies flow conservation. We can move one unit of flow per vertex in L, and so $|f| = |L| = |R| = n$, which is maximal.

Since all capacities are integers, there is another maximum flow f' such that $f'(e)$ is an integer for all $e \in E$. By the maximality condition, we must have $|f'| = n$. We know that f' can be translated into a bipartite matching of size n for G, i.e., a perfect bipartite matching, as desired.

Exercise 3

(Approximating the size of a maximum cut)

Throughout this course, we have run into the problem of finding cuts of a graph. Recall that a cut of a graph $G = (V, E)$ consists of two disjoint sets $S, T \subseteq V$ such that $S \cup T = V$. The size of the cut is the number of edges $(u, v) \in E$ such that $u \in S$ and $v \in T$, and the maximum cut of G is the cut of largest size in G.

For a long time, computer scientists have been interested in whether efficient algorithms exist for solving the so-called *MAX-CUT decision problem*:

Given a graph G and an integer k, is there a cut in G of size at least k?

In a beautiful work, Dick Karp showed that MAX-CUT is NP-complete. Roughly speaking, this means that no efficient algorithm exists for finding a maximum cut in a graph unless P $=$ NP! If you are curious, you can find Karp's original 21 combinatorial NP-complete problems at `https://en.wikipedia.org/wiki/Karp%27s_21_NP-complete_problems`.

Due to the difficulty of the MAX-CUT problem, a line of research has focused on developing efficient algorithms that *approximate* the size of the maximum cut. Let M denote the size of the maximum cut in a graph G. An algorithm outputs a *factor-k approximation* if it outputs a cut of size $m \geq M/k$.

The best known efficient approximation algorithm for MAX-CUT by Goemans and Williamson returns a cut of size at least $M/1.139$. In other words, this algorithm returns a factor-1.139 approximation of the maximum cut. Moreover, it has been shown that if the so-called *Unique Games Conjecture* is true, then this is the best approximation factor that an efficient algorithm can achieve! If you are curious, the original paper can be found at `http://www-math.mit.edu/~goemans/PAPERS/maxcut-jacm.pdf`.

Your goal in this exercise is to design a simple and efficient *randomized* algorithm that outputs a factor-2 approximation of the maximum cut. In other words, your algorithm should return a cut of size m such that its expected size $E[m]$ satisfies $E[m] \geq M/2$ for all graphs.

Hint: Consider the algorithm that chooses a cut (S, T) by flipping a coin for each vertex v to decide whether $v \in S$ or $v \in T$.

Model Answer

Fix a graph $G = (V, E)$. We create a set S by independently adding each vertex $v \in V$ to S with probability $1/2$. Let $T = V - S$. We now show that the expected size of the cut (S, T) is $|E|/2$. This implies the desired result as $|E| \geq M$, where M denotes the size of the maximum cut of G.

For each edge $e = (u, v) \in E$, let Z_e be the random variable such that $Z_e = 0$ if $u, v \in S$ or $u, v \in T$, and $Z_e = 1$ otherwise. Then, the size m of the cut (S, T) satisfies

$$m = \sum_{e \in E} Z_e.$$

We now compute the expected value of m, denoted by $E[m]$, as follows:

$$\begin{aligned} E[m] &= E\left[\sum_{e \in E} Z_e\right] \\ &= \sum_{e \in E} E[Z_e] \\ &= \sum_{e \in E} \Pr[Z_e = 1] \\ &= \sum_{e \in E} 1/2 \\ &= |E|/2. \end{aligned}$$

In the derivation above, the first equality holds by definition. The second equality holds via the linearity of expectation. The third equality is true because

$$E[Z_e] = 1 \cdot \Pr[Z_e = 1] + 0 \cdot \Pr[Z_e = 0] = \Pr[Z_e = 1].$$

Finally, the fourth equality holds because, if $e = (u, v)$, we have

$$\Pr[Z_e = 1] = \Pr[u \in S, v \notin S] + \Pr[u \notin S, v \in S] = 1/4 + 1/4 = 1/2.$$

As a final remark, observe that not only does the cut (S, T) have size $|E|/2$ in expectation, this reasoning also guarantees that every graph *must have* a cut of size at least $|E|/2$ (otherwise the expected value would be strictly smaller than $|E|/2$).
