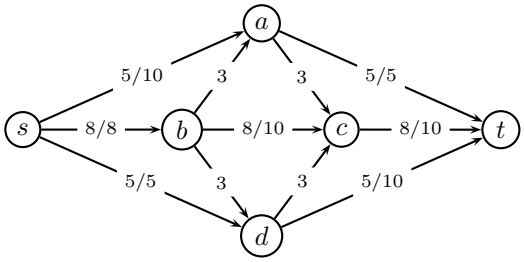
CS 624: Analysis of Algorithms

Assignment 6

Solutions

1. **Max-Flow-Min-Cut:** Given the following flow network on which an s-t flow has been computed. The capacity of each edge appears as a label on the edge, and the numbers in parentheses give the amount of flow sent on each edge. (Edges without parentheses – specifically, the four edges of capacity 3 – have no flow being sent on them.)



   (a) What is the value of this flow? Is this a maximum (s,t) flow in this graph?

   The value of the flow is 18, as can be seen by the sum of the flow on the edges leaving s or entering t.

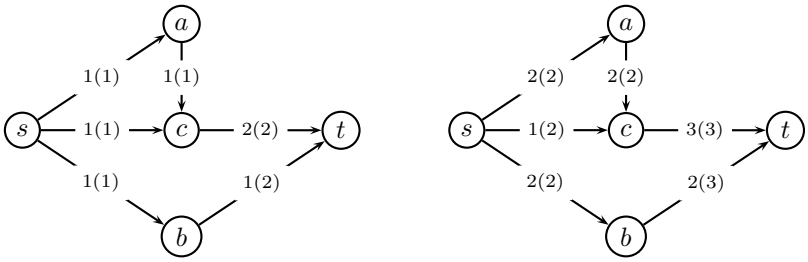
   (b) Find a minimum s-t cut in the flow network and also say what its capacity is.

   The minimum cut is $\{s, a\}$ and its capacity is 21. It is also the value of the maximum flow. It can be obtained by the following residual network below The augmenting paths (in no particular order) are:
   $s \to a \to t$ (cap. 5)
   $s \to b \to c \to t$ (cap. 5)
   $s \to a \to c \to t$ (cap. 3)
   $s \to d \to t$ (cap. 5)
   $s \to b \to d \to t$ (cap. 3).

   (c) Decide whether the following statement is true or false. If it is true, give a short explanation. If it is false, give a counter example:

   Given an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e ;and let (A, B) be a minimum $s-t$ cut with respect to these capacities $\{c_e | e \in E\}$. Now suppose we add 1 to every capacity; then (A, B) is still a minimum $s-t$ cut with respect to these new capacities $\{1 + c_e | e \in E\}$.

   The statement is false. See for example:



On the left the max-flow (and min-cut) are of value 3, and the cut is $\{s\}$ vs. $\{a, b, c, t\}$. On the right, after adding 1 to every capacity, the max flow and min-cut are 5, and the cut is now $\{s, c\}$ vs. $\{a, b, t\}$.

2. Exercise 1.1 (page 3 in the NP notes).

   Answer: The first graph has a Hamiltonian cycle, for example: $A \to B \to C \to D \to H \to G \to F \to E \to A$.

   The second graph has a Hamiltonian cycle, for example: $A \to B \to C \to I \to H \to G \to A$.

   The third graph doesn't have a cycle. For this we can use the lemma hinted in the question: If G is a graph that has a Hamiltonian cycle C, then every vertex of G is an endpoint of exactly 2 edges in C. Since C is a simple cycle (which it has to be, since it goes through every vertex once, see BST lecture notes), then it is of the form: $v_1 \to v_2 \to ... \to v_n \to v_1$, so that every vertex has one edge coming in and one going out.

   So, in the third graph, since A,C,G and I are of degree 2, both their edges must be part of a Hamiltonian cycle if there is one. Vertex B is connected to A and C on both sides, so it must be between them in the cycle. Same for F being between C and I, H being between I and G, and D being between A and G. So this forces us to have this following order: $A \to B \to C \to F \to I \to H \to G \to D \to A$. There is no way to get to E without going through a vertex more than once. Even if we start at E or get to it somehow, we can't escape going through one of the corner vertices, which will leave us stuck.

3. Exercise 2.1 (page 8)

   Answer: For a DNF expression to be satisfiable, it's enough that one of its clause is satisfiable. For a clause to be satisfiable, we need all the literals in it to have a True value. So a polynomial time algorithm to check the satisfiability of a DNF expression is as follows: For every clause, see if the clause doesn't contain a variable and its negation. If not, you can assign a True value to all the literals in this clause and the answer to the whole question is yes. Stop and return yes.

   Otherwise, move to the next clause and repeat the process. Only if all the clauses cannot be satisfied (That is, they contain a variable and its negation), return False.

   This is clearly linear in the size of the expression in the worst-case, so DNF-SAT is in P.

4. Exercise 3.6 (page 15)

   Answer:

   - If V1 is a vertex cover of G, then every edge touches at least one vertex in V1 by definition. If we look at V/V1 (all the vertices not in the VC), then if V1 is a vertex cover, no two vertices in V/V1 can share an edge, because for every edge, at least one end of it touches a vertex in V1. Hence, V/V1 is an independent set.

     Opposite direction: If V/V1 is an independent set, then no two vertices in the set share an edge. Therefore, every edge in the graph must touch at least one vertex from the remainder of the vertices, which are V1. This makes V1 a vertex cover by definition.

- If V2 is an independent set in G, then for each $u, v \in V2$ there is no edge (u,v). Then, in G^c, by definition, for each $u, v \in V2$ there is an edge (u,v), so V2 is a clique in G^c. The opposite direction is exactly the same.

5. In class (and in the notes) we showed that INDEPENDENT SET was NP-complete by showing that VC $\leq_P$ INDEPENDENT SET. Here is another way to do this, by showing directly that 3-SAT $\leq_P$ INDEPENDENT SET. I'm going to write most of the proof out. Your job will be to finish it up.

   Let $e = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ be a 3-SAT expression on n variables $v_1, v_2, \ldots, v_n$. (Remember that this means that each clause c_j is of the form $z_1^{(j)} \vee z_2^{(j)} \vee z_3^{(j)}$ where each $z_k^{(j)}$ is a *literal*—that is, it is either v_p or $\bar{v}_p$ for some p.)

   Our task is to construct a polynomial-time mapping that maps each such e into an undirected graph G such that

   $$G \text{ has an independent set of size } m \iff e \text{ is satisfiable.}$$

   We can assume that each clause c_j is made out of *distinct* variables, that is, no variable occurs twice in any one clause.

   First, let's take a simple example of a clause: $c = (v_1, \bar{v}_3, v_4)$. There are exactly 7 possible assignments of truth values to the variables v_1, v_3, and v_4 that give c the value T – in fact there are 8 possible assignments to the triplet $\langle v_1, v_3, v_4 \rangle$, and every one of these *except* $\langle \mathrm{F}, \mathrm{T}, \mathrm{F} \rangle$ makes c have the value T. Clearly this same reasoning applies to every clause, no matter how it is constructed.

   Now we are ready to construct the graph G. Each clause c_j in e will give rise to a group of 7 vertices in G. Each of these vertices will correspond to one of the 7 possible assignments of truth values to the variables in c_j that make c_j true.

   We do this for each of the clauses in e. This gives us m groups of 7 vertices each, for a total of $7m$ vertices in G.

   Now for the edges: We let the edges denote "interference", or "inconsistency". That is, we will put an edge between two vertices iff those two vertices assign opposite values to some variable.

   So for instance, every two vertices in any one of the groups has an edge between them. (That is, each group is a clique, although that fact is not important for us.) And there may also be other edges, going between vertices in different groups as well.

   Your task then is to show the following:

   (a) The transformation from e to G outlined above is a polynomial-time construction. (This is trivial really, but you have to say *something*.)

   Answer: We have m clauses. For each, we enumerate the seven assignments that satisfy the clause and construct seven vertices. Overall, $7m$ vertices, which is linear in the number of clauses. Then we go over all the vertices and draw an edge between any pair of opposing truth values. Within every group of seven vertices, we draw an edge, because each one assigns at least once an opposite value to some variable(s). Therefore, by definition, every such seven-tuple is a clique - this is a quadratic number of edges. Additionally, if we have recurring variables across different seven-tuples, we draw edges between them too if they assign opposite values to these recurring variables. Overall, there is no more than a quadratic number of edges in $7m$, and going over all pairs of vertices takes no more than quadratic time in the number of variables and clauses. Therefore, it is a polynomial construction.

   (b) e is satisfiable $\implies$ G has an independent set of size m.

   If e is satisfiable, every clause is satisfied, which corresponds to at least one vertex in each 7-tuple representing a clause that represents this assignment. Select this vertex from each

set of 7-tuple. We can be assured that none of these vertices has edges between them, because that would imply that the satisfying assignment assigns both T and F to a variable, which can't happen. Therefore, these vertices constitute an independent set of size m.

(c) G has an independent set of size $m \implies e$ is satisfiable.

If G has an independent set of size m, it must have exactly one vertex from each 7-tuple, because every 7-tuple is a clique and has all vertices mutually connected. An independent set like that means that there is a set of vertices whose assignments don't contradict one another. That is - that it doesn't assign both T and F to the same variable. Since every 7-tuple in each clause represent assignments that satisfy the clause, it means there is a valid truth assignment that satisfies the entire expression.

6. The SET-Packing problem is defined as follows: a set S together with a positive integer k. **Question:** Is there a collection C of k subsets of S that are pairwise disjoint, That is, no two subsets $c_i \in C$ and $c_j \in C$ share elements?

(a) Show that SET-Packing is in NP.

Answer: Given k subests of the set S, we can go over all pairs within the subset and make sure they don't share elements. It can be done is a time that's at most quadratic in the size of the sets and there is a quadratic number of such pairs. Overall, it is a polynomial verification.

(b) Show that SET-Packing is NP-complete using a reduction from the INDEPENDENT-SET problem. Don't forget to show the reduction is polynomial.

Answer: Given a graph G with n vertices and e edges that is an instance of INDEPENDENT-SET, build a set with n elements and v subsets such that each edge is an element and every subset is the set of edges that touch one of the vertices. Clearly, it is a polynomial construction, linear in the size of the graph, since we only have to go over all the edges and the adjacency relationship. If there is an independent set of size k, there is a collection of k subsets corresponding to these k vertices, and none share an element. In the opposite direction, if there is a collection of k pairwise disjoint subsets, they correspond to k vertices in G that don't share an edge, i.e. an independent set.