CSC373— Algorithm Design, Analysis, and Complexity — Spring 2018

Solutions for Assignment 3: NP and FILP

---

1. **[10pts] Interval Scheduling.** Suppose we are given a set of jobs $J(k)$ which start at time s_k and end at time f_k, where $s_k < f_k$ are non-negative integers for $k = 1, 2, \ldots, K$. We assume all the finish times are distinct. We wish to schedule a maximum size subset of these jobs such that no two scheduled jobs overlap in time (i.e., the open time intervals (s_j, f_j) and (s_k, f_k) do not intersect for any pair of jobs $J(j)$ and $J(k)$ that are scheduled). We represent such a maximal subset of jobs by the set of their indices, $S \subseteq \{1, 2, \ldots, K\}$.

   (a) **[5pts]** Consider the intersection graph for this problem $G = (V, E)$, where the vertices, say v_n, are in one to one correspondence with the jobs, $J(n)$ and there is an edge $(v_n, v_m) \in E$ iff $n \neq m$ and jobs $J(n)$ and $J(m)$ intersect (i.e., $(s_n, f_n) \cap (s_m, f_m) \neq \emptyset$). Show that this leads to a straight forward poly-time reduction of the interval scheduling problem using the independent set problem on G (i.e, IntervalSched$(\{J(n)\}_{n=1}^{N}) \leq_p$ searchIndepSet(G).

   (b) **[5pts]** Here we briefly consider what might be special about these intersection graphs to allow for a poly-time solution. Presumably the fact that each job runs in single time interval embedded in a common time axis is important for the existence of a polynomial time solution.

   Perhaps this is due to the fact that the jobs can be ordered, say by finish time. To investigate this, suppose we consider changing the above intersection graph G to a directed graph, where there is a directed edge $(v_j, v_k) \in E$ iff jobs $J(j)$ and $J(k)$ intersect and $f_j < f_k$. Show that the resulting graph G is a DAG (directed acyclic graph). Is there a simple greedy algorithm for the maximum independent set problem on a DAG, or is this problem also NP-hard?

Solution for Q1:

Soln Q1a: Given the input $J(k) = (s_k, f_k)$ for $k = 1, 2, \ldots, K$, define the size of the input to be $|s| = K + b$, where b is the maximum number of bits needed to specify each of the start and finish times, s_k and f_k. (**NOTE:** Here, as always, we define the input size $|s|$ to be something simple, for example, the sum of different monomials that determine the size of the input. Such a simple expression gives a $\Omega(|s|)$-bound on the input size. If the runtime or certificate length is bounded by a polynomial of this $|s|$, then they will be also be bounded by a polynomial in a more accurate estimate for input size.)

Define the reduction as follows. Let $V = \{v_k \mid k = 1, 2, \ldots, K\}$. Define $E = \{(v_j, v_k) \mid (s_j, f_j) \cap (s_k, f_k) \neq \emptyset\}$. Then E can be constructed in time $O(|s|^2)$, and it follows that the intersection graph $G = (V, E)$ can also be constructed in this time.

Next, the reduction makes one call, as follows.

$$V_S = searchIndepSet(G).$$

Finally, it returns the set $\{J(n) \mid v_n \in V_S\}$ as the solution to the interval scheduling problem given this input.

Due to the 1-1 correspondence between vertices v_k in G and jobs $J(k)$ we can simplify the notation and refer to S as the set of indices for which $V_S = \{v_n \mid n \in S\}$ and we define $J_S = \{J(n) \mid n \in S\}$ as the corresponding set of jobs returned by the above reduction.

This reduction is clearly poly-time, with just one call to searchIndepSet. The remaining issue is whether the returned solution is a maximum schedule for the given input problem. We begin with:

Claim 1. A subset of jobs $J_S = \{J(n) \mid n \in S\}$ is a feasible schedule (i.e., they will not mutually intersect) iff $V_S = \{v_n \mid n \in S\}$ is an independent set of G.

This follows from the construction of the intersection graph, specifically that $e = (v_j, v_k) \in E$ iff jobs $J(j)$ and $J(k)$ intersect.

We next use contradiction to show that the returned set $J_S = \{J(n) \mid !n \in S\}$ must be a maximum-sized feasible schedule. Suppose there exists a larger feasible schedule $\{J(n) | n \in T\}$ with $|T| > |S|$. Then, by Claim 1, V_T must be an independent set for G, and $|V_T| = |T| > |S| = |V_S|$. This contradicts V_S being a maximal independent set for G. ■

Soln Q1b: There is no poly-time algorithm for independent set on a DAG (unless $P = NP$). The reason for this is as follows.

Given any undirected graph $G = (V, E)$, we can always convert this to a DAG by choosing an appropriate direction for each edge. For example, you could pick any vertex $v \in V$, perform BFS to find the set of vertices connected to v, keeping the order in which these vertices were first visited by BFS. Directing the edges from vertices earlier in this order to later in this order cannot form a cycle. (Proof omitted.) Repeating this process for each connected component in G leads to a unique direction for all the edges in E. Say D is the set of these directed edges ($|E| = |D|$). Then $H = (V, D)$ is a DAG.

This conversion from G to H can be done in poly-time with respect to the size of the input graph G, say $|s| = |V| + |E|$. Therefore this leads to a poly-time reduction of searchIndepSet(G) (for a general undirected graph G) using searchIndepSetDAG(H) (for a DAG H). Therefore, if searchIndepSetDAG can be solved in poly-time, then so can searchIndepSet. But, since searchIndepSet(G) is NP-hard, so it follows that searchIndepSetDAG being in P implies $P = NP$.

Therefore we conclude that there is no poly-time greedy solution for searchIndepSetDAG, unless $P = NP$. In particular, the set of intersection graphs formed from interval scheduling problems must have some other special property that makes the independent set problem tractable for these graphs (see Tutorial Exercise 10, question 2).

2. **[10pts] Degree12Tree.** Using a polynomial reduction with one of the problems on slide 18 of the NP-complete lecture notes, show that the following problem is NP-complete:

   Degree12Tree(G): Given any (undirected) graph $G = (V, E)$, does there exist a subgraph $T = (V, F)$ of G (i.e., with $F \subseteq E$) such that, T is a spanning tree of G and the maximum degree of any vertex of T is 12?

   Note, the **degree** of a vertex v in $T = (V, F)$ is defined to be the number of edges in F that include v as one of their endpoints.

   Solution Q2. First, we show that degree12Tree is in NP. It is clearly a decision problem. Let $|s| = |V|+|E|$ denote the input size for G. Note that any spanning tree $T = (V, F)$ of G which has max degree equal to 12 would serve as a certificate t for the cases in which degree12Tree(G) is true. This length of this certificate satisfies $|t| \leq |s|$, so it is bounded by a polynomial in $|s|$.

   The appropriate certifier is given the input s and t and simply needs to check that $T = (V, F)$ has the appropriate properties. Specifically, it needs to check $F \subset E$ and the max degree of vertices v in T equals 12. To check that T is a spanning tree we can select any vertex v and use BFS to check that T is connected, and then ensure $|F| = |V| - 1$. These checks can all be done in poly-time with respect to the input size $|s|$. Therefore this is a poly-time certifier, and we conclude that degree12Tree is in NP.

   We want to show that HamCycle $\leq_p$ degree12Tree. Since HamCycle is NP-Complete this will show that degree12Tree is NP-Complete.

   You can use any of several possible definitions for a simple cycle in the definition of HamCycle. Here we require that a simple cycle has at least three edges, and no edge is used twice (in either direction). This means that no graph with less than three vertices can have a Hamiltonian cycle. In the following we assume that G has at least three vertices.

   Consider HamCycle(G) for some input graph $G = (V, E)$, where $|V| \geq 3$. Here we use the input size $|s| = |V| + |E|$. There are (at least) two general ways to approach this reduction. Both involve modifying G such that, for each $v \in V$, we add 10 or 11 new vertices, say $w(v, i)$ (see details further below), along with the corresponding edges $(v, w(v, i))$, for $i = 1, \ldots, N(v)$. The idea is that the added edges must be in any spanning tree and they will use up 10 or 11 edges of any degree 12 spanning tree. This leaves only one or two edges containing v in the orginal graph to be used for the spanning tree of this new graph. We can constrain these edges to form a path.

One approach to this problem is to consider any vertex v in G. Any Hamiltonian cycle in G must pass through v and include two edges (p, v) and (q, v) with $p, q \in V$, $p \neq q$. Given this observation, for each edge $e = (a, v) \in E$ (i.e., that has this v as an endpoint) we will test whether the graph G has a Hamiltonian cycle that includes this edge (a, v). (Choosing a low degree vertex v in G to begin with would make sense, as well as testing all but one edge through v, but neither of these refinements change the proof substantially.)

To make this test using degree12Tree, we build a new graph H as follows. We include all the vertices and edges of G, except for the one edge (a, v). For each $u \in V \backslash \{a, v\}$ we add 10 vertices $w(u, i)$, along with the 10 edges $(u, w(u, i))$, $i = 1, 2, \ldots, 10$. For each of the two vertices a and v, we add 11 new vertices $w(z, i)$, 11 new edges $(z, w(z, i))$, where $z = a$ or b. This forms the new graph H. Clearly H can be built from G in polynomial time with respect to the size of G (i.e., $|s| = |V| + |E|$).

Claim 1. H has a degree 12 spanning tree T iff G has a Hamiltonian cycle containing the edge (a, v).

Sketch of Pf. First we show that if G has a Hamiltonial cycle C containing the edge (a, v) then H has a spanning tree T, where every vertex which is not a leaf of T has degree 12. The argument here is simple, we remove the edge (a, v) from C and add the vertices $w(u, i)$ and edges to the resulting path. This forms a spanning tree of H where every non-leaf node is a vertex in C, and degree equal to 12. (We omit the details.)

Next we show that if H has a degree 12 spanning tree T then G has a Hamiltonian cycle containing the edge (a, v). Suppose T is such a spanning tree. It must contain all the vertices $w(u, i)$ (since it is spanning), moreover there must be a unique simple path P between any pair of such vertices (since it is a tree). This path can start or end at one of these added vertices $w(u, i)$, but it cannot visit such an added vertex in between the start and end vertex (since these added vertices have degree one, and the path P is assumed to be simple). This leads us to the following claim:

Claim 2. For any $x, y \in V$ there exists a unique simple path in T that does not visit any of the added vertices $w(u, i)$.

Next, consider the restriction of T to the original graph G, say $R = (V, F)$, where F is the set of edges in T with both endpoints in V. By construction (a, v) is not in F. By Claim 2, this graph R must be a spanning tree.

However, since we added 10 extra vertices $w(u, i)$ for each $u \in V \backslash \{a, v\}$, 11 extra vertices to a and v, and the maximum degree of T is equal to 12, then it follows that the maximum degree of R is two, and the degrees of a and v in R must both be one. That is, R must be a union of one or more simple paths with a and v two of the endpoints of these paths.

But we know R is a spanning tree, so R must be a single, simple, spanning path from v to a which does not include the edge (a, v). That is R is a Hamiltonian path between v and a, which does not contain the edge (a, v). Therefore, we can close this path with the edge (a, v) to form a Hamiltonian cycle. This completes the argument that if H has a degree 12 tree then G has a Hamiltonian cycle containing the edge (a, v). And this completes the proof of Claim 1, above. ■

The reduction now follows by testing each edge (a, v) that ends at v in the manner described above, calling degree12Tree(H) on the resulting graph. If any of these calls return true, then Claim 2 justifies the reduction also returning true (that is, G must have a Hamiltonian cycle). If none of these calls return true, then Claim 2 guarantees that there is no Hamiltonian cycle passing through any edge with v as an endpoint. That is, G cannot have a Hamiltonian cycle. This ensures the reduction returns false iff G does not have a Hamiltonian cycle.

Finally, the total overhead of constructing each of these graphs H is bounded by a polynomial in the input size $|s|$. Moreover, the number of graphs H that need to be built and the number of calls to degree12Tree are bounded by the degree of v, which in turn is bounded by $|s|$. Therefore, this is a poly-time reduction. ■

An alternative approach to testing each edge (a, v) is to begin by splitting a selected vertex, say $v \in V$, into two vertices, say a and b. Every edge ending at v, say $(u, v) \in E$, is then replaced by two edges (u, a) and (u, b). The two new vertices a and b are **not** directly connected by an edge (i.e., (a, b) is not in the new graph). We then add 10 new vertices $w(u, i)$ to each $u \in V \backslash \{v\}$, and 11 new vertices $w(z, i)$ to each of $z = a$ and $z = b$ in this split graph.

The claim is that this new graph has a degree12Tree iff the original graph G has a Hamiltonian cycle. The proof of this is similar to the above, where we consider the graph R which does not include any of the added vertices $w(u,i)$. Here again we need to show that the restricted graph R is of degree at most 2 and is a spanning tree. Moreover, we need to argue that if (x,a) and (y,b) are the two edges in R terminating at a and b, then $x \neq y$. (This will be impossible unless G has only two vertices, in which case it cannot have a Hamiltonian cycle.) It then follows that a Hamiltonian cycle for the original graph can be constructed by identifying a and b with the vertex v that we split. We omit the details of this second approach.

3. **[20pts] FILP.** Consider the feasibility problem for integer linear programming:

   FILP$(A,\vec{b})$: Given a $m \times n$ matrix A and a $m \times 1$ vector $\vec{b}$, where all elements of A and $\vec{b}$ are integers, consider the feasible set $F \subset \mathbb{Z}^n$ where $\vec{x}$ must satisfy

$$\begin{aligned} A\vec{x} &\leq \vec{b}, \\ \vec{x} &\geq \vec{0}. \end{aligned}$$

   That is, the decision problem FILP$(A,\vec{b})$ is to determine if there exist an integer-valued feasible solution $\vec{x} \in \mathbb{Z}^n$ of the above inequalities (i.e., is the set F defined above non-empty)?

   (a) **[2pts]** Show that FILP is in NP.

   (b) **[5pts]** Show that FILP is NP-complete by showing a poly-time reduction 3-Sat $\leq_p$ FILP. In this reduction you must have a one to one correspondence between the logical variables x_i^S in the 3-SAT formula and integer variables x_i^F in FILP, where x_i^S is true (false) iff $x_i^F = 1$ (0, respectively). Given the simplicity of this representation you can drop the superscripts S and F from these two sets of variables.

   (c) **[5pts]** Show that Set-Cover $\leq_p$ FILP with a similarly direct reduction. In this reduction you must use the variables $\vec{x}$ in FILP which are either 0 or 1, and $x_k = 1$ iff the k^{th} set in the Set-Cover problem has been selected.

   (d) **[3pts]** Use the reduction in part (c) to show that

$$\text{Set-Cover} \notin \text{co-NP} \implies \text{FILP} \notin \text{co-NP}.$$

   (e) **[5pts]** Consider the EqualThirds problem considered in Assignment 2, Question 4. That is, given a list of positive integers $P = (p_1, p_2, \ldots, p_n)$ we want to determine if it is possible to partition $I(n) = \{1, 2, \ldots, n\}$ into three mutually disjoint subsets, say $S_i \subset I(n)$ for $i = 1, 2, 3$, such that: a) $S_i \cap S_j = \emptyset$ for $i \neq j$; b) $I(n) = \cup_{i=1}^{3} S_i$; and c) the following equation holds:

$$\sum_{k \in S_1} p_k = \sum_{k \in S_2} p_k = \sum_{k \in S_2} p_k = \left[\sum_{k \in I(n)} p_k \right] /3. \tag{1}$$

   We wish to show EqualThirds is NP-complete (see Piazza for this claification). You must use a reduction for which $\vec{x}$ is a $2n \times 1$ vector with values in $\{0,1\}$, where $x_i = 1$ for $1 \leq i \leq n$ iff $i \in S_1$, and $x_{i+n} = 1$ for $1 \leq i \leq n$ iff $i \in S_2$.

   (f) **[0pts] Hard and Unmarked.** Show a polytime reduction of the Hamiltonian Cycle problem using FILP (i.e, show HamCycle $\leq_p$ FILP without going through another NP-complete problem).

Solution for Q3:

Soln Q3a: FILP$(A,\vec{b})$ is clearly a decision problem. Define the input size to be $|s| = m + n + B$, where A is $m \times n$, $\vec{b}$ is an m-vector, and B is the maximum number of bits required to specify any single element of A and $\vec{b}$. For convenience below, we define a_{max} to be the maximum absolute value of any element of A and b, so we can take

$$B = \log(a_{max}). \tag{2}$$

Claim 1. Any integer-valued feasible vector $\vec{x}$ can be specified by at most $O(n^2(log(n)+B))$ bits, which is bounded by a polynomial in $|s|$. Therefore, an integer-valued feasible vector $\vec{x}$ can serve as a certificate for this problem.

We prove Claim 1 further below. Given this claim, it follows that FILP is in NP since the conditions that $\vec{x}$ satisfies $A\vec{x} \leq \vec{b}$ and $\vec{x} \geq 0$ can be checked in polynomial time.

Proof of Claim 1. From the lecture slides, any vertex of the (real-valued) feasible set satisfies a linear equation of the form

$$Q(s)\vec{v}(s) = \vec{q}(s), \tag{3}$$

where s is a selection of row numbers from the problem constraints $A\vec{x} \leq \vec{b}$ and from $\vec{x} \geq 0$, and $Q(s)$ is nonsingular. Not all such solutions are vertices (they also need to satisfy $A\vec{v} \leq \vec{b}$ and $\vec{v} \geq 0$), but all vertices are a subset of the set of these points $\vec{v}(s)$ where $Q(s)$ nonsingular.

Since the feasible set is convex, any integer-valued feasible solution must be a convex combination of the feasible vertices. Therefore, the number of bits needed to represent each element of such a integer-valued feasible point is $O(log(v_{max}))$, where $v_{max} \geq |v_i(s)|$ for $i = 1, 2, \ldots, n$ and any row selection s for which $Q(s)$ is non-singular.

How big can v_{max} be? For any selection s, all the elements in $Q(s)$ and $q(\vec{s})$ are elements of A, I, $\vec{b}$, or zero, so they are integers that can be specified in B bits. Since $Q(s)$ is known to be nonsingular, we can write the elements of $\vec{v}$ using Cramer's Rule as

$$v_i = \frac{det(Q_i(s))}{det(Q(s))},$$

where $Q_i(s)$ is the the $n \times n$ matrix formed by replacing the i^{th} column of $Q(s)$ by the right hand side vector $\vec{q}(s)$. (Note that, since these matrices are integer-valued, and $det(Q(s)) \neq 0$, these values are all rational.)

Given that $Q(s)$ is integer valued, it follows that $det(Q(s))$ is an integer. Given that $det(Q(s)) \neq 0$, we have $|det(Q(s))| \geq 1$. Therefore

$$|v_i| = \frac{|det(Q_i(s))|}{|det(Q(s))|} \leq |det(Q_i(s))|. \tag{4}$$

We can now use the Leibniz formula for determinants to show that

$$|det(Q_i(s))| \leq n!(a_{max})^n$$

where a_{max} is as in eqn (2) above. From (4) we then find that

$$\begin{aligned} log(|v_i|) &\leq \log(|det(Q_i(s))|) \leq \log(n!) + n\log(a_{max}) \\ &\leq (n+1/2)\log(n) + n\log(a_{max}) + \log(e) + 1, \end{aligned} \tag{5}$$

where we have used the upper bound for $n!$ from Stirling's approximation.

Equation (5) ensures that the number of bits required to represent v_i is $O(n\log(n) + nB)$. Due to the convexity of the feasible set it follows that the elements of any feasible integer-valued solution also require at most $O(n\log(n)+nB)$ bits, and thus the number of bits required to represent any integer-valued feasible solution $\vec{x}$ is at most n times this number of bits. ■

Soln Q3b: Suppose Φ is any given 3-SAT clause. From Φ we wish to build the input A and $\vec{b}$ for one call to FILP$(A, \vec{b})$. We will arrange A and $\vec{b}$ such that that 3-SAT(Φ) is true iff FILP$(A, \vec{b})$ is true. In addition the construction of a suitable A and $\vec{b}$ will require polynomial time with respect to the size of the input for 3-SAT, namely $|\Phi|$, which we take to be $|\Phi| = n + m$ where n number of logical variables x_i^S and m is the number of clauses in Φ. Given all this, we will have shown that 3-SAT(Φ) $\leq_p$ FILP$(A, \vec{b})$, as desired.

Any given clause in the 3-SAT formula Φ has the form $L_{j_1} \vee L_{j_2} \vee L_{j_3}$, where L_{j_i} is a literal equal to $x_{j_i}^S$ or the logical negation $\bar{x}_{j_i}^S$. Recall that 3-SAT constrains the variables x_{j_i} in one class to be different from each other, so $j_i \neq j_k$ for $i \neq k$. In terms of the binary FILP variables discussed in the handout,

$x^F_{j_i} \in \{0,1\}$, we can express the logical negation as $1 - x^F_{j_i}$. This will be one iff $x^F_{j_i}$ is zero. For each clause we will use $n_i = 1$ to indicate that the i^{th} literal is negated, otherwise $n_i = 0$. It now follows that this 3-SAT clause is true iff

$$(-1)^{n_1} x^F_{j_1} + (-1)^{n_2} x^F_{j_2} + (-1)^{n_3} x^F_{j_3} \geq 1 - \sum_{i=1}^{3} n_i, \tag{6}$$

where we have moved the constants 1 for each negation over to the right hand side.

We build A and $\vec{b}$ by first iterating through the clauses in Φ. For each clause, we multiply (6) by minus one (to convert it to standard form) and include this linear constraint as one row of A and $\vec{b}$. That is the corresponding row of $A\vec{x} \leq \vec{b}$ is exactly -1 times (6). Next, we we need to append the constraints that $\vec{x} \leq 1$. This can be done by appending the $n \times n$ identity matrix I_n to the bottom of the current A and append the n-vector of all ones to the bottom of the current $\vec{b}$. This completes the construction of A and $\vec{b}$ (e.g., the matrix A is $(m+n) \times n$ and every element of A is in $\{0, \pm 1\}$, while the right hand side vector $\vec{b}$ has elements in $\{-1, 0, 1, 2\}$). This construction can clearly be done in poly-time in terms of $|\Phi| = n + m$.

Claim 2. An integer valued vector $\vec{x}^{\,F}$ is a feasible point of the ILP specified by (A, b) (i.e., $A\vec{x} \leq \vec{b}$ and $\vec{x} \geq 0$) iff the logical variables $\vec{x}^{\,S} \equiv (\vec{x}^{\,F} > 0)$ satisfy the CNF formula Φ.

We omit the details of this proof, which follow from the statement that (6) is satisfied by 0-1 variables x^F_j iff the corresponding clause in Φ is true.

Soln Q3c: Let (U, L, k) be the input for the SetCover problem, where U is a universe set, L is a list of subsets of U, and k is a positive integer. From (U, L, k) we wish to build the input A and $\vec{b}$ for one call to FILP$(A, \vec{b})$. We will build A and $\vec{b}$ such that SetCover(U, L, k) is true iff FILP$(A, \vec{b})$ is true. In addition, the construction of a suitable A and $\vec{b}$ will require polynomial time with respect to the size of this input for SetCover, namely $|(U, L, k)|$. Here we take $|s| = m + n + b$ where $m = |U|$, n is the length of the list L, and b is the number of bits required to specify k. (WLOG we can assume $0 \leq k \leq n$, so $b = O(log(n))$.) Given all this, we will have shown that 3-SAT $\leq_p$ FILP, as desired.

It is convenient to identify each of the elements of U with an integer $i = 1, 2, \ldots, m$, and denote each set in the list L, namely L_j for $j = 1, 2, \ldots, n$, to be a subset of $\{1, 2, \ldots, m\}$. This translation can be done in $O(nm)$-time.

We choose the ILP variable $\vec{x}$ to be an n vector, with each coefficient $x_i \in \{0, 1\}$, where $x_i = 1$ iff the subset L_i is to be included in the cover.

With this representation, for each of the m elements in U, the requirement that this element is covered by the L_j's for which $x_j = 1$ is equivalent to

$$\sum_{j=1}^{n} A_{i,j} x_j \leq -1 \tag{7}$$

where $A_{i,j} = -\delta(i \in L_j)$, where $\delta(p) = 1$ if p is true and $\delta(p) = 0$ otherwise.

Next we append A and $\vec{b}$ (below) by adding the constraint that at most k of the x_j's can be one, which is equivalent to

$$\vec{1}^{\,T} \vec{x} \leq k \tag{8}$$

Therefore we add a row of all ones to the A constructed so far, and we append the value k to the bottom of the current vector $\vec{b}$.

In addition to these constraints we must also constrain $x_i \leq 1$. To do that we append the bottom of A with an $n \times n$ identity matrix I, and append the bottom of the right hand side vector b with n ones. As a result A is a $(m + 1 + n) \times n$ matrix, with all its elements in $\{0, \pm 1\}$. Similarly, $\vec{b}$ is a $(m + 1 + n)$-vector with the first m elements being -1, the next equal to k, and the last n elements equal to 1.

This construction is clearly poly-time in $|s| = m + n + b$.

Claim 3. For the $A, \vec{b}$ constructed above, FILP$(A, \vec{b})$ is true iff SetCover(U, L, k).

We omit the proof of Claim 3.

Soln Q3d: We will show the contrapositve statement

$$\text{FILP} \in \text{co-NP} \implies \text{Set-Cover} \in \text{co-NP}.$$

Suppose FILP $\in$ co-NP. Then, by definition, for any input $s = (A, \vec{b})$, there exists a poly-time disqualifier, $DQ(s,t)$, and a polynomial p_D such that, if FILP(s) returns false then there exists an input t such that $|t| \leq p_D(|s|)$ and $DQ(s,t)$ returns true.

Given an instance of the SetCover problem, $s_S = (U, L, k)$, we can use the reduction in part (c) to generate an input $s_F = (A, \vec{b})$ for FILP. We refer to this transformation as $s_F = R(s_S)$. From part (c) this transformation runs in poly-time, and the size of s_F is bounded by a polynomial in $|s_S|$.

Note also that the reduction in part (c) simply returns the value of FILP$(A, \vec{b})$, which equals the value of SetCover(U, L, k). Therefore, if SetCover(U, L, k) returns false then FILP$(A, \vec{b})$ must also return false. Given the assumption that FILP is in co-NP, we know there must exist a certificate t, of poly-size wrt $s_F = (A, \vec{b})$, such that $DQ(s_F, t)$ is true.

Note the algorithm $DQ(R(s_S), t)$ runs in polytime with respect to s_S (since $DQ(s_F, t)$ runs in polytime in $|s_F|$, the transformation $R(s_S)$ runs in poly-time, and the size $|R(s_S)|$ is polynomial in s_S). Similarly, the length of the certificate t is polynomial in $|s_F|$, which is polynomial in terms of the size of $|s_S|$. Since the composition of polynomials is polynomial, it follows that $DQ(R(s_S), t)$ is a disqualifier for SetCover. Hence, we have SetCover $\in$ co-NP. ■

Soln Q3e: We will show EqualThirds $\leq_p$ FILP. Let the input for EqualThirds be $P = (p_1, p_2, \ldots, p_n)$. And let $T = \left[\sum_{k=1}^{n} p_k\right]/3$. (If this is not an integer then the reduction can return false.) We define the size of this input to be $|s| = n + b$ where b is the maximum number of bits needed to represent p_k for any k.

Define $\vec{x}$ to be a $2n$ vector with elements constrained by $x_i \in \{0, 1\}$. Here we wish to choose a representation such that, for $1 \leq i \leq n$, $x_i = 1$ iff $i \in S_1$, and $x_{i+n} = 1$ iff $i \in S_2$.

We can represent these constraints using FILP as follows. If one third of the total sum, namely T, is not an integer we return false. Otherwise, we begin constructing A and $\vec{b}$ row by row. The first two row of $A\vec{x} \leq \vec{b}$ correspond to

$$\begin{aligned} \sum_{i=1}^{n} p_i x_i &\leq T, \\ \sum_{i=1}^{n} (-p_i) x_i &\leq -T. \end{aligned}$$

That is, for all $i \in \{1, 2, \ldots, n\}$, $A_{1,i} = p_i$ and $A_{1,i+n} = 0$. The second row is just the negative of the first. Similarly the third and fourth rows of $Ax \leq \vec{b}$ are

$$\begin{aligned} \sum_{i=1}^{n} p_i x_{i+n} &\leq T, \\ \sum_{i=1}^{n} (-p_i) x_{i+n} &\leq -T. \end{aligned}$$

That is, for all $i \in \{1, 2, \ldots, n\}$, $A_{3,i} = 0$ and $A_{3,i+n} = p_i$. The fourth row of A is just the negative of the third. These constraints guarantee the the two subsets selected by $x_i = 1$ (or $x_{i+n} = 1$) for $i = 1, 2, \ldots n$ both sum exactly to T.

We still need to represent that these two sets are distinct, that is, no element p_i is used in both sets. To represent that we require that

$$x_i + x_{i+n} \leq 1,$$

for each $i = 1, 2, \ldots, n$. We can append these n rows to A (they have the form of a block matrix (I_n, I_n) where I_n is the $n \times n$ identity matrix), and append n entries with value 1 to the $\vec{b}$ that we are constructing.

The remaining rows of $A\vec{x} \leq \vec{b}$ correspond to the constraint that $\vec{x} \leq 1$. That is, the above rows of A are appended with a $2n \times 2n$ identity matrix I_{2n}, and the above rows of $\vec{b}$ are appended with $2n$ entries all equal to 1.

The resulting A matrix is $(4 + n + 2n) \times (2n)$, while A and $\vec{b}$ can be constructed in polytime given the input P.

It follows that FILP(A, b) is true iff EqualThirds(P) is true. We omit the proof.

Soln Q3f: One common mistake in developing this reduction is to only enforce the constraint that every vertex is used in exactly 2 edges of the subgraph that is meant to be the cycle. Such a constraint is easily arranged using FILP.

However, for any graph with at least 6 vertices, such a constraint allows for more than one connected subcycle to be a feasible solution of the resulting ILP problem. (For example, for six vertices, two simple cycles containing three vertices each satisfies this degree-2 constraint.)

Somehow the fact that the subgraph is degree two, connected, and spans all of the input vertices, must be encoded in FILP. See the paper Miller-Tucker-Zemlin formulation available from the library. (This paper is about the Travelling Salesperson Problem, but we can simply toss away the objective function to reduce the HamCycle problem using FILP.)

For still more information see The bad and the good-and-ugly: formulations for the TSP, by Gábor Pataki.