We use the result of the previous problem. We first classify all nodes as upstream, downstream, or central. If there exist any central nodes, then by the definition of "central," this implies that there is not a unique minimum cut.

On the other hand, if there are no central nodes, then the partition of the nodes into upstream and downstream defines a cut (A, B). Suppose there were some minimum cut (A', B') other than (A, B). Then either there is a node $v \in A - A'$, contradicting the definition of the upstream nodes, or there is a node $v \in B - B'$, contradicting the definition of the downstream nodes. Thus (A, B) is the unique minimum cut in this case.

[1] ex310.28.237

(a, b) We define a graph $G = (V, E)$ with source s, vertices $v_1, \ldots, v_d$ for each day, vertices $w_1, \ldots, w_k$ for each person, and sink t. There is an edge of capacity 1 from s to each v_i, an edge of capacity 1 from v_i to each w_j with $p_j \in S_i$, and an edge of capacity $\lceil \Delta_j \rceil$ from w_j to t. We know there is a feasible fractional flow in this graph of value d, obtained by assigning a flow of value of $\frac{1}{|S_i|}$ to each edge (v_i, w_j), and flow values to all other edges as implied by the conservation condition. Thus there is a feasible integer flow, and the flow values on the edges of the form (v_i, w_j) define a fair driving schedule in the following way: p_j drives on day i if and only if $f(v_i, w_j) = 1$. This also gives a polynomial-time algorithm to compute the schedule.

[1] ex879.304.520

If we put a capacity of 1 on each edge, then by the integrality theorem for maximum flows, there exist k edge-disjoint x-y paths if and only if there exists a flow of value k. By the max-flow min-cut theorem, this latter condition holds if and only if there is no x-y cut (A, B) of capacity less than k.

Now suppose there were a y-x cut (B', A') of capacity strictly less than k, and consider the x-y cut (A', B'). We claim that the capacity of (A', B') is equal to the capacity of (B', A'). For if we let $\delta^-(v)$ and $\delta^+(v)$ denote the number of edges into and out of a node v respectively, then we have

$$\begin{aligned} c(A', B') - c(B', A') &= |\{(u,v) : u \in A', v \in B'\}| - |\{(u,v) : u \in B', v \in A'\}| \\ &= |\{(u,v) : u \in A', v \in B'\}| + |\{(u,v) : u \in A', v \in A'\}| - \\ &\quad\ |\{(u,v) : u \in A', v \in A'\}| - |\{(u,v) : u \in B', v \in A'\}| \\ &= \sum_{v \in A} \delta^+(v) - \sum_{v \in A} \delta^-(v) \\ &= 0. \end{aligned}$$

It follows that $c(A', B') < k$ as well, contradicting our observation in the first paragraph. Thus, every y-x cut has capacity at least k, and so there exist k edge-disjoint y-x paths.

[1] ex52.743.508

Let a^* be the earliest arrival time of any job, and d^* the latest deadline of any job. We break up the interval $I = [a^*, d^*]$ at each value of any a_j, d_j, t_i, or t'_i. Let the resulting sub-intervals of I be denoted $I_1, I_2, \ldots, I_r$, with $I_i = [s_i, s'_i]$. Note that $s'_i = s_{i+1}$ in our notation; we let $q_i = s'_i - s_i$ denote the length of interval I_i in time steps. Observe that the set of processors available is constant throughout each interval I_i; let n_i denote the size of this set. Also, the set of jobs that have been released but are not yet due is constant throughout each interval I_i.

We now construct a flow network G that tells us, for each job j and each interval I_i, how much time should be spent processing job j during interval I_i. From this, we can construct the schedule. We define a node u_j for each job j, and a node v_i for each interval I_i. If $I_i \subseteq [a_j, d_j]$, then we add an edge (u_j, v_i) of capacity q_i. We define an edge from the source s to each u_j with capacity ℓ_j, and define an edge from each v_i to the sink t with capacity $n_i q_i$.

Now, suppose there is a schedule that allows each job to complete by its deadline, and suppose it processes job j for z_{ji} units of time in the interval I_i. Then we define a flow with value ℓ_j on the edge (s, u_j), value z_{ji} on the edge (u_j, v_i), and sufficient flow on the edge (v_i, t) to satisfy conservation. Note that the capacity of (v_i, t) cannot be exceeded, since we have a valid schedule.

Conversely, given an integer flow of value $\sum_j \ell_j$ in G, we run job j for $z_{ji} = f(u_j, v_i)$ units of time during interval I_i. Since the flow has value $\sum_j \ell_j$, it clearly saturates each edge (s, u_j), and so u_j will be processed for ℓ_j units of time, as required, if we can guarantee that all jobs j can really be scheduled for z_{ji} units of time during interval I_i. The issue here is the following: we are told that job j must receive z_{ji} units of processing time during I_i, and it can move from one processor to another during the interval, but we need to avoid having two processors working on the same job at the same point in time. Here is a way to assign jobs to processors that avoids this. Let $P_1, P_2, \ldots, P_{n_i}$ denote the processors, and let $y_j = \sum_{r<j} z_{ri}$. For each $k = y_j + 1, y_j + 2, \ldots, y_{j+1}$, we have processor $\lceil k/q_i \rceil$ spend the $(k - q_i \lfloor k/q_i \rfloor)^{\text{th}}$ step of interval I_i working on job j. Since $\sum_j z_{ji} \leq n_i q_i$, each job gets a sufficient number of steps allocated to it; and since $z_{ji} \leq q_i$ for each j, this allocation scheme does not involve two processors working on the same job at the same point in time.

[1] ex304.152.546

Build a flow network G with vertices s, v_i for each $x_i \in X$, w_j for each $y_j \in Y$, and t. There are edges (s, v_i) for all i, (w_j, t) for all j, and (v_i, w_j) iff x_i and y_j have appeared together in a movie. All edges are given capacity 1. Consider a maximum s-t flow f in G; by the integrality theorem, it consists of a set $\{R_1, \ldots, R_k\}$ of edge-disjoint s-t paths, where k is the value of f. (Note that this is simply the construction we used to reduce bipartite matching to maximum flow.)

Suppose the value of f is n. Then each time player 1 names an actress x_i, player 2 can name the actor y_j so that (v_i, w_j) appear on a flow path together. In this way, player 1 must eventually run out of actresses and lose.

On the other hand, suppose the value of f is less than n. Player 1 can thus start with an actress x_i so that v_i lies on no flow path. Now, at every point of the game, we claim the vertex for the actor y_j named by player 2 lies on some flow path R_t. For if not, consider the first time when w_j does not lie on a flow path; if we take the sequence of edges in G traversed by the two players thus far, add s to the beginning and t to the end, we obtain an augmenting s-t path, which contradicts the maximality of f. Hence, each time player 2 names an actor y_j, player 1 can name an actress x_ℓ so that (x_ℓ, y_j) appear on a flow path together. In this way, player 2 must eventually run out of actors and lose.

[1] ex559.764.819

The problem is in $\mathcal{NP}$ because we can exhibit a set of k customers, and in polynomial time is can be checked that no two bought any product in common.

We now show that *Independent Set* $\leq_P$ *Diverse Subset*. Given a graph G and a number k, we construct a customer for each node of G, and a product for each edge of G. We then build an array that says customer v bought product e if edge e is incident to node v. Finally, we ask whether this array has a diverse subset of size k.

We claim that this holds if and only if G has an independent set of size k. If there is a diverse subset of size k, then the corresponding set of nodes has the property that no two are incident to the same edge — so it is an independent set of size k. Conversely, if there is an independent set of size k, then the corresponding set of customers has the property that no two bought the same product, so it is diverse.

[1] ex640.690.659

The problem is in NP since, given a set of k counselors, we can check that they cover all the sports.

Suppose we had such an algorithm $\mathcal{A}$; here is how we would solve an instance of *Vertex Cover*. Given a graph $G = (V, E)$ and an integer k, we would define a sport S_e for each edge e, and a counselor C_v for each vertex v. C_v is qualified in sport S_e if and only if e has an endpoint equal to v.

Now, if there are k counselors that, together, are qualified in all sports, the corresponding vertices in G have the property that each edge has an end in at least one of them; so they define a vertex cover of size k. Conversely, if there is a vertex cover of size k, then this set of counselors has the property that each sport is contained in the list of qualifications of at least one of them.

Thus, G has a vertex cover of size at most k if and only if the instance of *Efficient Recruiting* that we create can be solved with at most k counselors. Moreover, the instance of *Efficient Recruiting* has size polynomial in the size of G. Thus, if we could determine the answer to the *Efficient Recruiting* instance in polynomial time, we could also solve the instance of *Vertex Cover* in polynomial time.

[1] ex195.705.667

4-Dimensional Matching is in NP, since we can check in $O(n)$ time, using an $n \times 4$ array initialized to all 0, that a given set of n 4-tuples is disjoint.

We now show that *3-Dimensional Matching* $\leq_P$ *4-Dimensional Matching*. So consider an instance of *3-Dimensional Matching*, with sets X, Y, and Z of size n each, and a collection C of ordered triples. We define an instance of *4-Dimensional Matching* as follows. We have sets W, X, Y, and Z, each of size n, and a collection C' of 4-tuples defined so that for every $(x_j, y_k, z_\ell) \in C$, and every i between 1 and n, there is a 4-tuple (w_i, x_j, y_k, z_ℓ). This instance has a size that is polynomial in the size of the initial *3-Dimensional Matching* instance.

If $A = (x_j, y_k, z_\ell)$ is a triple in C, define $f(A)$ to be the 4-tuple (w_j, x_j, y_k, z_ℓ); note that $f(A) \in C'$. If $B = (w_i, x_j, y_k, z_\ell)$ is a 4-tuple in C', define $f'(B)$ to be the triple (x_j, y_k, z_ℓ); note that $f'(B) \in C$. Given a set of n disjoint triples $\{A_i\}$ in C, it is easy to show that $\{f(A_i)\}$ is a set of n disjoint 4-tuples in C'. Conversely, given a set of n disjoint 4-tuples $\{B_i\}$ in C', it is easy to show that $\{f'(B_i)\}$ is a set of n disjoint triples in C. Thus, by determining whether there is a perfect 4-Dimensional matching in the instance we have constructed, we can solve the initial instance of *3-Dimensional Matching*.

[1] ex420.972.30

Path Selection is in NP, since we can be shown a set of k paths from among $P_1, \ldots, P_c$ and check in polynomial time that no two of them share any nodes.

Now, we claim that *3-Dimensional Matching* $\leq_P$ *Path Selection*. For consider an instance of *3-Dimensional Matching* with sets X, Y, and Z, each of size n, and ordered triples $T_1, \ldots, T_m$ from $X \times Y \times Z$. We construct a directed graph $G = (V, E)$ on the node set $X \cup Y \cup Z$. For each triple $T_i = (x_i, y_j, z_k)$, we add edges (x_i, y_j) and (y_j, z_k) to G. Finally, for each $i = 1, 2, \ldots, m$, we define a path P_i that passes through the nodes $\{x_i, y_j, z_k\}$, where again $T_i = (x_i, y_j, z_k)$. Note that by our definition of the edges, each P_i is a valid path in G. Also, the reduction takes polynomial time.

Now we claim that there are n paths among $P_1, \ldots, P_m$ sharing no nodes if and only if there exist n disjoint triples among $T_1, \ldots, T_m$. For if there do exist n paths sharing no nodes, then the corresponding triples must each contain a different element from X, a different element from Y, and a different element from Z — they form a perfect three-dimensional matching. Conversely, if there exist n disjoint triples, then the corresponding paths will have no nodes in common.

Since *Path Selection* is in NP, and we can reduce an NP-complete problem to it, it must be NP-complete.

(Other direct reductions are from Set Packing and from Independent Set.)

[1]ex529.979.546

The problem is in $\mathcal{NP}$ because we can exhibit a set of bidders, and in polynomial time is can be checked that no two bought bid on the same item, and that the total value of their bids is at least B.

We now show that *Independent Set* $\leq_P$ *Diverse Subset*. Given a graph G and a number k, we construct a bidder for each node of G, and an item for each edge of G. Each bidder v places a bid on each item e for which e is incident to v in G. We set the value of each bid to 1. Finally, we ask whether the auctioneer can accept a set of bids of total value $B = k$.

We claim that this holds if and only if G has an independent set of size k. If there is a set of acceptable bids of total value k, then the corresponding set of nodes has the property that no two are incident to the same edge — so it is an independent set of size k. Conversely, if there is an independent set of size k, then the corresponding set of bidders has the property that their bids are disjoint, and their total value is k.

[1] ex617.432.555