1.

a. Yes, Minimum Cut algorithm runs in polynomial time. It can be solved without referring to vertex cover.

b. Unknown, because it would resolve the question of whether P = NP.

Independent Set can be solved in polynomial time so in case Independent Set ≤ p Minimum Cut, then Independent Set can be solve in polynomial time and is NP-complete. NP-complete problem can be solved in polynomial time if P = NP.

2.

Step 1: We need to construct a set H given size k from the given graph G

Step 2: take in H to black-box B, repeat step 1-2 with power time.

Step 3: Access black-box B with power time. Check the set is independent-set or not, it’s easy to iterate all vertexes to check it’s independent or not, this would take polynomial time.

Step 4: If there’s no set H is independent-set, then the result is ‘no’, otherwise ‘yes’

3.

This problem can be reduce to graph 3-coloring problem.

Step 1: Create a graph G=(V,E) that vertexes are talks and participants, the edges are connected by talk interests of participants. Easy to see no edge among participants, so assume all participant vertexes are only 1 color.

Step 2: Solve this problem by graph 3-coloring algorithm. Sort the grade of talk vertexes in order decreasing → V’

Step 3: Start from i=1, Iterate all vertexes in V’ and coloring i-color to vertexes that has no edge to another i-color vertex

Step 4: Check if all vertexes in V’ are colored (It’s easy to check by iterating all participant vertexes and check their edges , this would take in polynomial time), then program ends, result is coloring number (except participant color). Otherwise, to step 5.

Step 5: Remove the colored-vertexes, start from i=i+1 and come back to step 2.

That means consider all possible cases, this would take exponential time.

If the result <= k, then the answer is ‘yes’, otherwise ‘no’

Since to check if there are talks in same time is running in polynomial time, so the problem ∈ NP.

Also this problem can be reduced to graph 3-coloring (in NP-complete), so this problem has to be a NP-complete.

4.

This problem can be reduced to logic-CSP problem.

Step 1: start from i=1, add xi = 1

Step 2: Check if collection of clauses over unnegated variables are satisfied or not. This would take in polynomial time. If yes, then the result is i, otherwise to step 3

Step 3: remove xi , start from i=i+1, set xi =1 and come to step 1,

That means we consider all possible set of X(x1,x2,...xn)

If the result <= k, then the answer is ‘yes’, otherwise ‘no’

Absolutely, check if result is satisfied or not take polynomial time, so problem ∈ NP.

Also this problem can be reduced to logic-CSP (in NP-complete), so this problem has to be a NP-complete.

5.

This problem can be reduced to Steiner tree problem.

Given G=(V,E,w) W⊂ V

Step 1: Construct graph G’=(V,F,w’) that w’(u,v) is the minimum distance from u→v (using Floyd-Warshall)

Step 2: Consider set S ⊂ V-W, find minimum-spanning-tree T’(Kruskal) in G’

Check if all vertexes distance in all set S are satisfied <= B. This can be done by finding minimum-spanning-tree, would take polynomial time.

Step 3: Construct Steiner tree T from T’ by replacing edges between in G’ by the path in G. The vertexes ∈ T but not ∈ W is Steiner vertices.

The result is ‘yes’ if the set number < k and satisfy no two in set S > B

Since to check if satisfy the condition run in polynomial time, this problem also can be reduced to Steiner tree, so this problem ∈ NP-complete.

6.

It takes polynomial time to check an assignment satisfy exactly all but one of the clauses.

Since The 3-CNF-SAT problem is a NP-Complete problem, so this problem also is a NP-problem.