Algorithm Design and Analysis (Fall 2022) Assignment 6

Xiangyuan Xue (521030910387)

1. For any given path P with length $\frac{|V|}{2}$, we can examine whether P is a simple path in G within linear time. Hence, the problem is in NP.

   To show its NP-completeness, we reduce the problem from HamiltonianPath. Given an instance $G = (V, E)$ of HamiltonianPath, corresponding instance $G' = (V', E')$ of the problem is constructed as follow: $V' = V \cup V^0$, $|V| = |V^0| = \frac{1}{2}|V'|$, $E' = E$. In other words, G' is built by adding $|V|$ isolated vertices to G.

   Since the isolated vertices will not form any path, there exists a Hamiltonian path in G if and only if there is a simple path with length exactly $\frac{|V'|}{2}$ in G', which completes the reduction.

2. For any given schedule P, we can examine within linear time whether each job i begins no earlier than r_i and ends no later than d_i, whether no overlapping happens, and whether every job is done within a consecutive time interval. Hence, the problem is in NP.

   To show its NP-completeness, we reduce the problem from SubsetSum+. Given an instance $(S = \{x_1, x_2, \dots, x_n\}, k)$ of SubsetSum+, corresponding instance $J = \{(r_i, d_i, t_i)\}$ of the problem is constructed as follow: Let $s = \sum_{i=1}^{n} x_i$. For each x_i, create a job i where $(r_i, d_i, t_i) = (0, s + 1, x_i)$. Another job $n + 1$ is also created where $(r_{n+1}, d_{n+1}, t_{n+1}) = (k, k + 1, 1)$.

   On one hand, if there exists a subset $T \subseteq S$ whose numbers sum up to exactly k, corresponding jobs can be fully assigned to the time interval $[0, k]$. The remaining jobs can be assigned to the time interval $[k, s + 1]$ since it is long enough. On the other hand, if there exists a schedule where all the jobs can be finished, no idle time should exist within the time interval $[0, k]$. Otherwise, there will not be enough time for the remaining jobs. Corresponding numbers from S sum up to exactly k. Hence, $(S = \{x_1, x_2, \dots, x_n\}, k)$ is a yes instance if and only if $J = \{(r_i, d_i, t_i)\}$ is a yes instance, which completes the reduction.

3. Let we skip (a) and directly prove the conclusion in (b).

   For any given solution $\{x_1, x_2, \dots, x_n\}$, we can examine whether it is feasible within polynomial time. Hence, the problem is in NP.

   To show its NP-completeness, we reduce the problem form VertexCover. Given an instance $(G = (V, E), k)$ of VertexCover, corresponding instance $\{C_1, C_2, \dots, C_m\}$ of the problem is constructed as follow: Let $\{x_1, x_2, \dots, x_n\}$ be the integer decision variables where $n = |V|$. For each edge $(u, v) \in E$, build an inequality constraint that $x_u + x_v \geq 1$, which requires that (u, v) is covered at least once. For each vertex $u \in V$, build an inequality constraint that $0 \leq x_u \leq 1$, which indicates whether u is selected. To ensure that the size of vertex cover is exactly k, we add a constraint that $\sum\limits_{u \in V} x_u = k$, which can be reformulated as an inequality that $k \leq \sum\limits_{u \in V} x_u \leq k$. Note that these inequalities can be easily translated into standard form, so they are equivalent to the constraints in an integer linear program.[1]

On one hand, if there exists a vertex cover of size k in G, corresponding decision variables will form a feasible solution. On the other hand, if there is a feasible solution, the variables imply a vertex cover of size k in G. Hence, $(G = (V, E), k)$ is a yes instance if and only if $\{C_1, C_2, \ldots, C_m\}$ is a yes instance, which completes the reduction.

Reconsider the conclusion in (a). For any given solution $\{x_1, x_2, \ldots, x_n\}$, we can examine whether it is feasible and whether $\sum\limits_{i=1}^{n} c_i x_i \geq k$ holds within polynomial time, so the problem is in NP. Since deciding the existence of feasible solution is already NP-complete and the problem is even more complex, we can conclude that the problem is also NP-complete.

6. (a) It takes me about 8 hours to finish this assignment.
   (b) I prefer a 5/5 score for its difficulty.
   (c) I have no collaborators. Papers and websites referred to are listed below.

References

[1] Wikipedia. Integer Program. https://en.wikipedia.org/wiki/Integer_programming