The problem is in NP since, given a set of k intervals, we can check that none overlap. Suppose we had such an algorithm  $\mathcal{A}$ ; here is how we would solve an instance of 3-Dimensional Matching.

We are given a collection C of ordered triples  $(x_i, y_j, z_k)$ , drawn from sets X, Y, and Z of size n each. We create an instance of *Multiple Interval Scheduling* in which we conceptually divide time into 3n disjoint *slices*, labeled  $s_1, s_2, \ldots, s_{3n}$ . For each triple  $(x_i, y_j, z_k)$ , we define a job that requires the slices  $s_i$ ,  $s_{n+j}$ , and  $s_{2n+k}$ .

Now, if there is a perfect tripartite matching, then this corresponds to a set of n jobs whose slices are completely disjoint; thus, this is a set of n jobs that can be all be accepted, since they have no overlap in time. Conversely, if there is a set of jobs that can all be accepted, then because they must not overlap in time, the corresponding set of n triples consists of completely disjoint elements; this is a perfect tripartite matching.

Hence, there is a perfect tripartite matching among the triples in C if and only if our algorithm  $\mathcal{A}$  reports that the constructed instance of *Multiple Interval Scheduling* contains a set of n jobs that can be accepted to run on the processor. The size of the *Multiple Interval Scheduling* instance that we construct is polynomial in n (the parameter of the underlying 3-Dimensional Matching instance).

Another way to solve this problem is via Independent Set: here is how we could use an algorithm  $\mathcal{A}$  for Multiple Interval Scheduling to decide whether a graph G = (V, E) has an independent set of size at least k. Let m denote the number of edges in G, and label them  $e_1, \ldots, e_m$ . As before, we divide time into m disjoint slices,  $s_1, \ldots, s_m$ , with slice  $s_i$  intuitively corresponding to edge  $e_i$ . For each vertex v, we define a job that requires precisely the slices  $s_i$  for which the edge  $e_i$  has an endpoint equal to v. Now, two jobs can both be accepted if and only if they have no time slices in common; that is, if and only if they aren't the two endpoints of some edge. Thus one can check that a set of jobs that can be simultaneously accepted corresponds precisely to an independent set in the graph G.

 $<sup>^{1}</sup>$ ex346.976.515