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

Solutions for Tutorial Exercise 9: NP-completeness of K-D Matching

1. **3D Matching.** We consider the following types of 3D matching problems.

   partial3DM: Given three distinct sets X, Y, and Z, with $|X| = |Y| = |Z| = n$, a set of triples $T \subseteq X \times Y \times Z$, and an integer k, does there exist a subset of triples $C \subseteq T$ of size $|C| \geq k$ such that no two distinct elements $C_i, C_j \in C$ have any element in common (i.e., if $C_i = (C_{i,1}, C_{i,2}, C_{i,3})$ and $C_j = (C_{j,1}, C_{j,2}, C_{j,3})$ are distinct triples in C then, for each $p = 1, 2, 3$, we have $C_{i,p} \neq C_{j,p}$)?

   perfect3DM: The input to this problem is similar to partial3DM except no integer k is provided. Instead, the question is whether there exists a set of triples $C \subseteq T$ such that $|C| = n$ and no two distinct elements $C_i, C_j \in C$ have any of their three elements in common? (That is, the matching is perfect in the sense that each element of X, Y, or Z is covered by exactly one triple in C.)

   Note Wikipedia defines the "3D matching problem" to be the problem partial3DM above, while the Kleinberg and Tardos text defines it to be perfect3DM. Moreover, in your answers below you can use the fact that 3-SAT $\leq_p$ perfect3DM, which is proved in the Kleinberg and Tardos text.

   (a) Show partial3DM is in NP.

   (b) Show perfect3DM $\leq_p$ partial3DM.

   (c) Given all the above results (including those quoted from Kleinberg and Tardos) can you conclude that partial3DM is NP-complete? Explain.

   Solution for Q1:

   1a **Soln.** Note that partial3DM(s) is a decision problem, where the input s refers to X, Y, Z, T, and k. We can define the size of the input to be $|s| = n + |T|$ (where $|T| \leq n^3$). Here, as is common, we are taking liberties with the actual number of bits necessary to specify the length of the input, knowing that we are only ever going to need to argue about runtimes (or sizes) being bounded by a polynomial in $|s|$. Note that, if $k > n$, the answer is no, so we only need to worry about the case $k \leq n$, which is bounded by $|s|$.

   Suppose partial3DM(X, Y, Z, T, k) is true. Then there must exist a subset $C \subseteq T$ such that $|C| = k$ and distinct elements of C do not intersect. We can take the certificate t to specify this set of triples C.

   In more detail than necessary: We could represent any such solution as a list of, say, indices i_j for the elements of T that are included in C, for $j = 1, 2, \ldots, k$. Each of these indices can be specified in $O(\log(|T|)$ bits. So $|t| \in O(k \log(|T|) \subset O(|s| log(|s|)$. Therefore the length of the certificate, $|t|$, is bounded by $O(p_C(|s|))$, where $p_C(x) = x^2$.

   Finally, we define the certifier, $C(s, t)$, to be an algorithm that simply checks the condition that no two elements in C intersect. This can be done in time $O(k^2) \subset O(|s|^2)$, so the certifier is poly-time.

   1b **Soln.** Note perfect3DM(X, Y, Z, T) = partial3DM(X, Y, Z, T, k) for $k = n$. That is, perfect3DM is a special case of partial3DM.

   1c **Soln.** Since 3-SAT is NP-complete, and 3-SAT $\leq_p$ perfect3DM $\leq_p$ partial3DM, it follows that partial3DM is NP-hard. By (1a) it is also in NP, thereforepartial3DM is NP-complete.

2. Consider partial2DM, which is the same as the 3D version except there are only two sets, X and Y, and T consists of a set of pairs $T \subseteq X \times Y$.

   (a) Is partial2DM in NP? Explain.

   (b) Show partial2DM $\leq_p$ SetPack, where SetPack is the set packing problem considered in the previous tutorial.

   (c) Is partial2DM NP-complete? Explain.

Solution for Q2:

2a **Soln.** This is similar to (1a) above. Note that partial2DM is a decision problem. Define $|s| = n + |T|$. In the case partial2DM(X, Y, T, k) is true, we can define $C \subseteq T$ to be an example solution with $|C| = k$. This corresponds to a certificate, t, of polynomial size wrt $|s|$. Then the certifier $C(s, t)$ need only check the solution provided by t. This takes at most $O(k^2)$ time, where $k \leq n \leq |s|$. Therefore, partial2DM has a poly-time certifier $C(s, t)$ with a certificate length bounded by $p_C(|s|) = |s|^2$ (see Q1a above). Therefore partial2DM is in NP.

2b **Soln.** Given any input (X, Y, T, k) for the problem partial2DM, define $U \equiv X \times Y$. Then partial2DM(X, Y, T, k) is a special case of SetPack(U, T, k), and partial2DM(X, Y, T, k) is true iff SetPack(U, T, k) is true. This completes the reduction, which clearly can be done in poly-time.

2c **Soln.** Notice that partial2DM(X, Y, T, k) can be solved in poly-time by using an s-t flow (it can be directly mapped to the bipartite matching problem). Therefore partial2DM$(X, Y, T, k) \in P$. So, if partial2DM was in fact NP-complete, then $P = NP$. Currently, we believe $P \neq NP$, so all we can say at this point is that it is unlikely that partial2DM is NP-complete.

3. Consider partial4DM, which is the same as the 3D version except there are now four sets, W, X, Y and Z, while T consists of a set of 4-tuples $T \subseteq W \times X \times Y \times Z$.

   (a) Is partial4DM in NP? Explain.

   (b) Is partial4DM NP-complete? Explain.

Solution for Q3:

3a **Soln.** It is a decision problem and, in the cases where the decision is yes, a solution provides a suitable short certificate that can be checked in polynomial time.

3b **Soln.** Consider the NP-complete problem partial3DM, and suppose we are given a suitable input (X, Y, Z, T, k). Define $W = X$, and T' to be the set of 4-tuples

$$T' = \{(w, x, y, z) \mid (x, x, y, z) \in T\}.$$

Then it follows that partial4DM(W, X, Y, Z, T', k) is true iff partial3DM(X, Y, Z, T, k). Therefore we have shown partial3DM$(X, Y, Z, T, k) \leq_p$ partial4DM(W, X, Y, Z, T, k), and it follows that partial4DM(W, X, Y, Z, T, k) is NP-hard. Since it is also in NP (see (3a) above), it follows that partial4DM(W, X, Y, Z, T, k) is NP-complete.

4. **(Harder.)** Show partial3DM $\leq_p$ SAT by using an encoding of the constraints for 3D matching in terms of a CNF formula. Use the binary variables x_i, where x_i is true iff the i^{th} triple in T is to be included in the set C. (Note that we are asking simply for a reduction to SAT, not to 3-SAT.)

Soln Problem 4. Consider the problem partial3DM and any suitable input (X, Y, Z, T, k). Define the binary variables x_i, for $i = 1, 2, \ldots, m$ where $m = |T|$, such that x_i is true iff the i^{th} triple in T is to be included in the set C. WLOG we assume that $k \leq |T| = m$.

We next encode the constraint that, for $i \neq j$, x_i and x_j cannot both be true if the two triples T_i and T_j intersect (that is, for some $p \in \{1, 2, 3\}$, we have $T_{i,p} = T_{j,p}$). This is done by including the constraint $\neg(x_i \wedge x_j)$, which is logically the same as the disjunctive clause $(\bar{x}_i \vee \bar{x}_j)$. The eventual SAT formula will include such a clause for every pair of T_i and T_j which intersect. There are at most $O(|T|^2)$ disjunctive clauses, and each such clause can be computed in constant time.

The conjunction of all the clauses computed so far will guarantee that the set of triples selected according to x_i being true correspond be a set C of triples which satisfy the pairwise non-intersection property of the 3D matching problem. The remaining constraint we need to encode is that $|C| \geq k$. That is, for $i = 1, 2, \ldots, m$, at least k of the logical variables x_i must be true.

One (unsuitable) attempt to constrain at least k of the x_i to be true is as follows. Consider all disjunctive combinations of $m-k+1$ distinct variables x_i. If we assume at least k variables x_i are true (over $i = 1, \ldots, m$) then, given any selection of $m - k + 1$ such variables, at least one of the variables in the selection must be true. This can be expressed as a simple disjunction over all the selected variables. Moreover, we can obtain a

necessary and sufficient condition for at least k variables to be true by taking the conjunction over **all these disjunctive clauses**. However there will be $\binom{m}{m-k+1}$ such disjunctive clauses or, equivalently, $\binom{m}{k-1}$ such clauses (see binomial coefficients).

While this approach does give a SAT formula that is satisfiable if and only if partial3DM(X, Y, Z, T, k) should return true, this reduction algorithm does not necessarily run in poly-time. Indeed, the number of clauses in the resulting SAT formula produced is not bounded by a polynomial in $|s|$. Specifically, $\binom{m}{k-1}$ grows like $\Omega(2^m/m^{1/2})$ for $k-1 = m/2$ (look up lower bounds for binomial coefficients and Stirling's formula). The number of clauses therefore can grow exponentially with m (and exponentially with $|s|$), so this is not a poly-time reduction.

How can we more efficiently encode the condition that at least k of the binary variables must be true? Consider a $k \times m$ assignment matrix A, where the i, j element of A is a logical variable $a_{i,j}$. The idea is that the i^{th} row of A effectively specifies which of the m logical variables is to be considered the i^{th} logical variable, say $x_{j(i)}$, that must be true (according to A). We require A to have exactly one variable in each row that is true, and at most one variable in each column of A is true. We will show how to write SAT constraints for A further below.

For the moment, assume we can build any such A. Then, for each $j = 1, 2, \ldots, m$, we consider the implication $(a_{1,j} \vee \ldots \vee a_{k,j}) \implies x_j$. Such an implication forces x_j to be true whenever the j^{th} column of A has at least one $a_{i,j}$ which is true. Moreover, the constraints on A force k different columns to have one true value, and therefore at least k different x_j's will be forced to be true. Finally, although this is not critical, by using the logical implication we will allow additional x_j's to be true as well.

Note that $(a_{1,j} \vee \ldots \vee a_{k,j}) \implies x_j$ is equivalent to $x_j \vee (\neg(a_{1,j} \vee \ldots \vee a_{k,j}))$, which in turn is equivalent to the conjunction of the 2-clauses $(x_j \vee \bar{a}_{i,j})$ for all $1 \leq i \leq k$. Thus there are m such implications, and each of these implications can be written as the conjunction of k clauses of size 2, so there are $O(mk)$ such clauses.

In order to write the constraints on A note that we can enforce at least one value in the i^{th} row of A to be true by including the single clause $(a_{i,1} \vee \ldots \vee a_{i,m})$. Next, to enforce the constraint that exactly one variable in the i^{th} row is true we only need to add the constraint that at most one variable in $\{a_{i,j}\}_{j=1}^{m}$ is true. This is equivalent to the condition that $\neg(a_{i,p} \wedge a_{i,q})$ is true or, equivalently, $(\bar{a}_{i,p} \vee \bar{a}_{i,q})$ is true, for all $p \neq q$ with $1 \leq p, q \leq m$. There are $O(m^2)$ such clauses for each of the k rows of A. Finally, the constraint that at most one variable in each column of A is true generates $O(k^2)$ more clauses of the form $(\bar{a}_{p,j} \vee \bar{a}_{q,j})$ for each column $j = 1, 2, \ldots, m$.

Since $k \leq m$, the total number of clauses is $O(m^3) \subset O(|s|^3)$, and these can all be generated in poly-time with respect to $|s|$.

Finally, we need to argue that the resulting SAT formula is satisfiable iff the original decision problem partial3DM(X, Y, Z, T, k) is true. This argument relies on the three major properties of this reduction. First, the binary variable x_i is true iff the triple $T_i \in C$. Secondly, no pair of x_i and x_j can be true if $T_i \cap T_j \neq \emptyset$. Third, the clauses forming the construction and use of the matrix A are true iff at least k of the x_i are true. We omit the details of this argument.