Recitation 6: NP-Completeness (continued)

1 Bin Packing

Optimization($a$): Given $n$ items with sizes $a = (a_1, a_2, \ldots a_n)$ such that $a_i \in (0, 1]$, what is the minimum number of bins $k$ such that there exists a $k$-partition of $[n]$ denoted by $S_1, S_2, \ldots S_k$ such that for all $j, \sum_{i \in S_j} a_i \leq 1$.

Search($a, k$): Given $n$ items with sizes $a = (a_1, a_2, \ldots a_n)$ such that $a_i \in (0, 1]$, find a $k$-partition of $[n]$ denoted by $S_1, S_2, \ldots S_k$ such that for all $j, \sum_{i \in S_j} a_i \leq 1$ (if one exists).

Decision($a, k$): Given $n$ items with sizes $a = (a_1, a_2, \ldots a_n)$ such that $a_i \in (0, 1]$, does there exist a $k$-partition of $[n]$ denoted by $S_1, S_2, \ldots S_k$ such that for all $j, \sum_{i \in S_j} a_i \leq 1$?

1.1 Reduce Bin-Pack-Opt to Bin-Pack-Search

Run Bin-Pack-Search for decreasing values of $k$ beginning with $k = n$ until Bin-Pack-Search($a, k$) returns a packing. Alternatively use binary search.

1.2 Reduce Bin-Pack-Search to Bin-Packing

1. If Bin-Packing($a, k$) == No, return null.

2. While $n > k$:
   (a) For every pair of items ($i, j$):
      i. Assuming $i < j$, modify the set of items $a$ by merging items $i$ and $j$. Remove item $j$ and add size $a_j$ to the size of item $i$.
      ii. If Bin-Packing($a, k$) == Yes, then we know that there exists a $k$-partition where items $i$ and $j$ are in the same partition.
      iii. Otherwise, if Bin-Packing($a, k$) == No, then items $i$ and $j$ must be in different partitions for every feasible solution. Thus, we add back item $j$ and subtract the size $a_j$ from item $i$.

3. There will be only $k$ items left, thus a trivial partition works. To recover the original $k$-partition, we keep track of which items were merged together along this process.

A loose analysis shows that this requires at most $n^3$ calls to Bin-Packing, which is polynomial in the input.
2 Prove Max2SAT is NP-Complete: Reducing from Clique

Clique$((G, k))$: Given graph $G = (V, E)$ and integer $k$, is there a set of vertices $U \subseteq V$ with $|U| \geq k$ that form a complete graph ($k$-clique)?

Max2SAT$((C, X, k))$: Given a CNF formula consisting of clauses $C = \{c_1, c_2, \ldots, c_n\}$ and literals $X = \{x_1, x_2, \ldots, x_k\}$ such that each clause involves exactly two literals, does there exist an assignment of TRUE/FALSE to the literals such that at least $k$ clauses are satisfied?

2.1 Show Max2SAT $\in$ NP

To prove this, we need to prove there exists a polytime verifier. Given any “yes” input (CNF formula), the “witness/certificate” is an assignment of TRUE/FALSE to the literals that satisfies at least $k$ clauses. A polytime verifier can simple evaluate each clause of the CNF formula given the assignment, and verify that at least $k$ of them are satisfied.

2.2 Show Max2SAT $\in$ NP-Hard:

1. To prove this, we reduce Clique to Max2SAT. Given an input $(G, k)$ to Clique, we will create an input $(C, X, k')$ to Max2SAT such that “yes” instances of Clique map to “yes” instances of Max2SAT, and “no” instances of Clique map to “no” instances of Max2SAT.

   (a) In designing the CNF formula, we use literals $x_1, x_2, \ldots, x_n$ to represent the $n$ vertices in the graph. We want to design clauses that act as constraints to enforce that $x_1 = \text{TRUE}$ corresponds to vertex 1 being chosen in the clique.

   (b) Recall that a set of vertices $U$ is a clique if for all $i, j \in U, (i, j) \in E$. Equivalently (by the contrapositive), $U$ is a clique if for all $i, j \in V$ such that $(i, j) \notin E$, either $i \notin U$ or $j \notin U$. We will use these constraints to design the clauses in our CNF formula.

   (c) Therefore, for every “non-edge” $(i, j) \notin E$, we have a clause $(\neg x_i \lor \neg x_j)$. This means that for every non-edge, at least one of the endpoints must not be in the clique.

   (d) However, these clauses could also be satisfied by setting all literals $x_i$ to FALSE. In order to encourage choosing cliques with more vertices, for every vertex $i$ we introduce the clauses $(x_i \lor z) \land (x_i \land \neg z)$, where $z$ is a new literal. In order to satisfy these two clauses, $x_i$ must be true.

   (e) Choose $k' = \text{number of non-edges} + |V| + k$.

Therefore, the input to Max2SAT is defined by

- literals $X = V \cup \{z\},$
- clauses $C = \{(\neg x_i \lor \neg x_j) \text{ for all non-edges } (i, j)\} \cup \{(x_i \lor z) \land (x_i \land \neg z) \text{ for all vertices } i\}$,
- and $k' = |\text{non-edges}| + |V| + k$. 

2
2. Show that if Clique is “yes”, then Max2SAT is “yes”. Given a clique $U \subseteq V$ in graph $G$ such that $|U| \geq k$, we can use an assignment such that for all $i \in U$, $x_i$ is TRUE, for all $i \notin U$, $x_i$ is FALSE, and $z$ is TRUE. This is an assignment that satisfies at least $k'$ clauses in our CNF formula. First, for all non-edges $(i,j)$, the clause $(\neg x_i \lor \neg x_j)$ is satisfied because $U$ is a clique. For all $i \in U$, both $(x_i \lor z)$ and $(x_i \land \neg z)$ are satisfied. For all $i \notin U$, $(x_i \lor z)$ is satisfied while $(x_i \land \neg z)$ is not satisfied. Therefore, the number of satisfied clauses is $|\text{non-edges}| + 2|U| + |V \setminus U| = |\text{non-edges}| + |V| + |U| \geq k'$.

3. Show that if Max2SAT is “yes”, then Clique is “yes”. Given an assignment for the literals $X$ such that at least $k'$ clauses are satisfied, we will show that we can find a $k$-clique in the original graph. If for all $(i,j) \notin E$, $(\neg x_i \lor \neg x_j)$ is satisfied, then the literals that are set to TRUE form a clique in graph $G$ that has size $\geq k$ (by construction). However, if the current assignment does not correspond to a clique, then we will show that we can modify the assignment to find a clique of size at least $k$. For every clause $(\neg x_i \lor \neg x_j)$ that is not satisfied, we change $x_i$ to FALSE (doesn’t matter which one we pick), thus satisfying this clause. This will cause one of $(x_i \lor z)$ or $(x_i \land \neg z)$ to become not satisfied. In addition, $x_i$ may also appear in other “non-edges” clause. Therefore, the net change is that the number of clauses satisfied can only increase or stay the same by this modification. We continue to choose unsatisfied “non-edge” clauses and modifying the assignment in this way until all the “non-edge” clauses are satisfied. Note that the number of unsatisfied “non-edge” clauses is monotonically decreasing in each modification. Therefore, we obtain an assignment which corresponds to a clique (since all “non-edge” clauses are satisfied). In addition the clique has size at least $k$ because the number of satisfied clauses is still at least $k'$, since every modification does not decrease the number of satisfied clauses.

3 Reducing from ZOE to Subset Sum

**ZOE:** Given an $n \times n$ matrix $Q$ consisting of $0-1$ entries, does there exist a $0-1$ vector $x = (x_1, \ldots, x_n)$ such that $Qx = 1$ (where $1$ denotes the all ones vector)?

**Subset Sum:** Given a set of integers $A = \{a_1, \ldots, a_m\}$, and an integer $s$, does any non-empty subset $A' \subseteq A$ sum to $s$?

Reduction: When $x$ is a $0-1$ vector, the result of $Qx$ is simply the sum of the columns of $Q$ for which the corresponding entry in $x$ is nonzero. Therefore, we consider each column of $Q$ as an integer written in base $n+1$, such that these correspond to the integers $\{a_1, \ldots, a_n\}$. Similarly we consider $1$ to be an integer written in base $n+1$ and set $s$ equal to that value. Then solving for a $0-1$ vector $x$ such that $Qx = 1$ is equivalent to finding a subset of $\{a_1, \ldots, a_n\}$ which adds up to $s$. We use base $n+1$ in order to get around the issue of carries. Therefore, it is clear that by this mapping, the subset $A'$ that sums to $s$ corresponds exactly to a $0-1$ vector $x$ such that $Qx = 1$. 

3