Homework4

Qinyun Song

1 2-SAT

1. *Proof.* Let us consider the truth table.

x_1	x_2	$x_1 \vee x_2$	$\neg x_1 \rightarrow x_2$
true	true	true	true
true	false	true	true
false	true	true	true
false	false	false	false

From the table we can see that, $x_1 \vee x_2$ is logically equivalent to the clause $(\overline{x_1} \rightarrow x_2)$. □

2. If there is a problematic x_i, it means that in the original formula, there exist two clauses like this: $(x_i \vee x_i) \wedge (\overline{x_i} \vee \overline{x_i})$). This formula can never be satisfied. If there is no problematic x_i, we can simply assign each l_i as true or false for each edge so that they will never contradict. Otherwise there must be a problematic variablebecause the only way to contradict is to have a problematic variable. If we have the assignment of each variable, we can apply it to the original formula and it can satisfy the formula. If the formula is satisfiable, we can apply the assignment to the graph. Since the only condition to show contradict is to have a problematic variable and the formula is satisfied, there cannot be any problematic variable in the graph. So after all, the formula ϕ is satisfiable if and only if there is no problematic variable.

2 Beating exhaustive search

1. *Proof.* For the clause $l1_1 \vee l_2 \vee l_3$, there are eight possible combinations.

l_1	l_2	l_3	How to represent this situation
T	T	T	$l_1 = TRUE$
T	T	F	$l_1 = TRUE$
T	F	T	$l_1 = TRUE$
T	F	F	$l_1 = TRUE$
F	T	T	$l_1 = FALSEandl_2 = TRUE$
F	T	F	$l_1 = FALSEandl_2 = TRUE$
F	F	T	$l_1 = FALSEandl_2 = FALSEandl_3 = TRUE$
F	F	F	N/A

So from the table we can see that, the three recurses can cover all the situation. For last line when all of them are false, in this case we don't need to find an assignment anymore since this clause equals false. So that situation is ignored. □

2. As stated in the problem, the formula ϕ has n variables and $m = poly(n)$ caluses. $T(n, m)$ denote the time used by the algorithm. For n, m, the running time needed for three variables must be no less than the clause contain at most two literals. So we can only consider the situation that there are exactly three variables. In the situation, we will try at most three recurses. For the first one, we assign TRUE to l_1 and try to calculate the remaining. The time needed for the remaining part is $T(n-1, m)$ because we eliminate one variable l_1 now. If the first recursive fails, we will enter the second one. In this situation, we assign $l_1 = FALSE$ and $l_2 = TRUE$. So the time needed for this recursive is $T(n-2, m)$ because we assigned values for two variables. If this fails too, we will try the third recursive. So the time needed for this is $T(n-3, m)$ since all three l_1, l_2, l_3 are assigned some value. So for the worst case, the running time $T(n, m)$ has the equation that

$$T(n, m) \leq T(n-1, m) + T(n-2, m) + T(n-3, m) + poly(n)$$

The last item in the equation $poly(n)$ is the running time needed to process assigning value to the variables and similar stuff. The most basic situation of this recursion is

$$T(1, m) = 2$$

Because we can simply assign true or false to the only variable. Suppose for $n \leq k$, the maximum running time $T(n, m) = O((1.9)^n)$. Then for $n = k$, we have

$$\begin{aligned} O(T(n, m)) &\leq O(T(n-1, m)) + O(T(n-2, m)) + O(T(n-3, m)) + O(poly(n)) \\ &\leq O(T(n-1, m) + T(n-2, m) + T(n-3, m) + poly(n)) \\ &\leq O((1.9)^{n-1} + (1.9)^{n-2} + (1.9)^{n-3} + cn) \end{aligned} \tag{1}$$

Using the math method, let $x = O(T(n, m))$, we can get that $x \approx 1.9$ and $x \leq 1.9$. So the running time can be no bigger than $O((1.9)^n)$.

3 Sudoku as SAT

Define variable x_{ijk} means that for the entry in $i-th$ row and $j-th$ column, the number we fill in is k. If it equals one, means that we fill in the number. Otherwise we won't choose to fill it using k. So now we have several constrains.

1. We need to make sure that for each entry, we only pick one number to fill in. So we have:

$$x_{ij1} \vee x_{ij2} \vee \cdots \vee x_{ijn} = 1 \tag{2}$$
$$x_{ij1} + x_{ij2} + \cdots + x_{ijn} = 1 \tag{3}$$

2. We need to make sure that in each row, we use all n numbers. Thus for each number k, we have

$$x_{i1k} \vee x_{i2k} \vee \cdots \vee x_{ink} = 1 \tag{4}$$
$$x_{i1k} + x_{i2k} + \cdots + x_{ink} = 1 \tag{5}$$

3. We need to make sure that for each column, numbers from 1 to n occurs exactly once. Then for each number k, we have

$$x_{1jk} \vee x_{2jk} \vee \cdots \vee x_{njk} = 1 \tag{6}$$
$$x_{1jk} + x_{2jk} + \cdots + x_{njk} = 1 \tag{7}$$

4. We need to make sure that for each *block*, there are integers from 1 to n. Then for number k, for the *block* in $l_i - th$ row and $l_j - th$ column (here the row and the column are divided according to the block instead of entries), we first find the position of all the entries and defined as $p_1, \cdots, p_n$. They are easy to compute. For example, for p_1, it equals $((i-1) \times \sqrt{n} + 1, (j-1) \times \sqrt{n} + 1)$. Then for the nine entries, we have:

$$x_{p_1 k} + x_{p_2 k} + \cdots + x_{p_n k} = 1 \tag{8}$$
$$x_{p_1 k} \vee x_{p_2 k} \vee \cdots \vee x_{p_n k} = 1 \tag{9}$$

As shown in the class, the equations in the form as

$$x_1 + \cdots + x_n = 1 \tag{10}$$
$$x_1 \vee \cdots \vee x_n = 1 \tag{11}$$

can be transformed into a SAT. So for all the x_{ijk} entries, and the above equations, we can transform them into a SAT problem. Once we find the value for each x_{ijk} using the SAT algorithm, if $x_{ijk} = 1$, fill in the entry in the $i - th$ row and $j - th$ column as the number k.

4 A basic reduction

1. *Proof.* For a 3-SAT problem, suppose it has n variables and m clauses. Then for each clause c_i, there are three different situations.

   (a) $c_i = l_1$ In this situation, convert it into

   $$l_1 \leq 1 \tag{12}$$
   $$-l_1 \leq -1 \tag{13}$$

   When l_1 is x_i, replace l_1 by x_i. Otherwise replace l_1 by $1 - x_i$ and then move all the constant numbers to the right side.

   (b) $c_i = l_1 \vee l_2$. In this situation, convert it into

   $$l_1 + l_2 \leq 2 \tag{14}$$
   $$-l_1 - l_2 \leq 1 \tag{15}$$

   Replace l_1, l_2 using the corresponding x variables using the above rule.

   (c) $c_i = l_1 \vee l_2 \vee l_3$. In this case, we can have:

   $$l_1 + l_2 + l_3 \leq 3 \tag{16}$$
   $$-l_1 - l_2 - l_3 \leq -1 \tag{17}$$

   Also replace l_i by x variables using the same rule.

   So after all, we can see that, the 3-SAT problem can be converted into a ILP problem using several functions polynomial in n. Since 3-SAT is NP-hard and ILP is at least as hard as 3-SAT, we can see that ILP is NP-hard. □

2. I think it is NP-complete. The problem itself is NP because we can validate the solution in polynomial time but we cannot come up with an answer in polynomial time. Since it is also NP-hard, it is thus NP-complete.

3. I don't think it has a polynomial time algorithm. Because first it can be converted to a 3-SAT problem in polynomial time. Secondly, as stated above, the 3-SAT problem can be converted to this problem. So I think it is NP-complete. Thus it has no polynomial time algorithm.

5 Hardness amplification

1. *Proof.* Suppose the MIS set in the graph G is a set of vertices $S = \{x_1, ..., x_k\}$. Then in the graph $G \otimes G$, there are k^2 vertices in its MIS set with vertices (x_i, x_j) where $i = 1 \cdots k, j = 1 \cdots k$. Because if there exist a pair $(a, b), (c, d)$ where $a, b, c, d \in S$ and (a, b) is connected with (c, d), then it means that in the original graph G, there is an edge from a to c or from b to d. These both will violate the assumption that the vertices in S is a set of MIS. So the points chosen from $G \otimes G$ are in the MIS of it. On the other hand, if there are more than k^2 vertices in $G \otimes G$, suppose there is a vertice (a, b) and since it has more than k^2 vertices, either a or b is not in the MIS of original graph G. Then there must exist a node in the set such that it is connected with either a or b, say c. This will result that in the constructed graph, the poiint (a, c) or (b, c) is connected with (a, b) and they are chosen too. This will result in the contradiction. So the size of MIS of graph $G \otimes G$ can only be exactly k^2. □

2. *Proof.* For the constructed graph $G \otimes G$, if there is a 4-approximation algorithm for MIS of it, suppose the size of the MIS is k, then the size of MIS of G is $\sqrt{k}$. And the size of G is $\sqrt{|G \otimes G|}$. Since $k/|G \otimes G| \geq 1/4$, we can see that $\sqrt{k}/|sqrt|G \otimes G| \geq 1/2$. So we can see that there must be a 2-aproximation algorithm for the original graph G. □

6 Power of randomness

1. *Proof.* First of all, consider the simplest case. That is, x and y only differ at position i. Then if r_i equals zero, the output is wrong. If r_i is one, the output is true. So the probability that $< x, r > \neq < y, r >$, modulo 2 is at least $\frac{1}{2}$.
If x and y differs at k positions $l_1, \cdots, l_k$, then for the random string r, it has $\frac{1}{2}$ probability with that bit equals 1. So if the product module 2 is different, we need the bit of r equals one with odd number of these different positions. The probability that there are odd number of positions is $\frac{1}{2}$. So with probability at least $\frac{1}{2}$, we can get different product modulo 2. □

2. From above we can see that, if we pick one random string and do the test, the probability of the error is $\frac{1}{2}$. Then for the second time, the probability of error is $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. So after 10 rounds, the probability of error is $\frac{1}{2^{10}} = \frac{1}{1024} < \frac{1}{1000}$. So after 10 rounds, the probability of error is less than $\frac{1}{1000}$.