CS 577: Introduction to Algorithms Fall 2020

Homework 10 Solutions

Instructor: Dieter van Melkebeek TA: Nicollas Mocelin Sdroievski

Problem 1

A CNF formula is *monotone* if the literals in each clause are either all positive or all negative. The problem of MONO-SAT is the restriction of CNF-SAT to monotone formulas: Given a monotone CNF formula, does it have a satisfying assignment? Show that MONO-SAT is NP-hard.

Given a formula φ in CNF, our approach will be to substitute each occurrence of a negated variable x by a new non-negated variable ℓ, while adding some clauses to force ℓ to be equal to $\overline{x}$. More formally, let φ be an instance of CNF-SAT with m clauses and n variables. We construct a monotone Boolean formula φ' as follows. For each variable x_j that appears negated in some clause, we add the variable ℓ_j. Then, we replace every occurrence of $\overline{x_j}$ by ℓ_j. Finally, we add the clauses $(\ell_j \vee x_j)$ and $(\overline{\ell_j} \vee \overline{x_j})$ for each ℓ_j. The formula φ' is monotone by construction, and we claim that it is satisfiable if and only if φ is satisfiable.

Claim 1. *There is a satisfying assignment for φ if and only if there exists a satisfying assignment for φ'.*

Proof. We consider both directions

$\Longrightarrow$ Assume there is a satisfying assignment X for φ. Set each variable of φ' that was also originally in φ according to X, and set $\ell_j = \overline{x_j}$ for all j. This satisfies all of the new clauses that were added to construct φ' as well as the clauses that were originally in φ, since after taking into consideration that $\ell_j = \overline{x_j}$, these clauses are identical to the original ones which are satisfied by X.

$\Longleftarrow$ Assume there is a satisfying assignment X' for φ'. Set every variable of φ according to X', ignoring the extra variables. Because φ' is satisfied by X', it must be the case that X' sets $\ell_j = \overline{x_j}$, otherwise one of $(\ell_j \vee x_j)$ or $(\overline{\ell_j} \vee \overline{x_j})$ would not be satisfied by X'. This implies that the assignment we have constructed satisfies φ, since again after taking into consideration that $\ell_j = \overline{x_j}$, these clauses are identical to the ones in φ' which are satisfied by X.

□

The formula φ' has a total of at most $2n$ variables and $m + 2n$ clauses, and constructing it can be done in polynomial time. Because CNF-SAT reduces to MONO-SAT in polynomial time and CNF-SAT is NP-hard, MONO-SAT is also NP-hard.

Alternate solution based on resolution Let φ be an instance for CNF-SAT with m clauses and n variables. We construct a monotone Boolean formula φ' as follows: For each clause $C_i = x_1 \vee x_2 \vee \cdots \vee x_s \vee \overline{y_1} \vee \overline{y_2} \vee \ldots \overline{y_t}$ in φ, add two clauses $C'_{2i-1} = x_1 \vee x_2 \vee \cdots \vee x_s \vee z$ and $C'_{2i} = \overline{z} \vee \overline{y_1} \vee \overline{y_2} \vee \ldots \overline{y_t}$ to φ', where z is a fresh variable. The Boolean formula φ' is monotone by construction.

Those of you familiar with resolution will notice that C_i is the resolvent of C'_{2i-1} and C'_{2i}:

$$x_1 \vee x_2 \vee \cdots \vee x_s \vee \overline{y_1} \vee \overline{y_2} \vee \ldots \overline{y_t} \equiv (x_1 \vee x_2 \vee \cdots \vee x_i \vee z) \wedge (\overline{z} \vee \overline{y_1} \vee \overline{y_2} \vee \ldots \overline{y_t}),$$

Now, we prove that the property we were aiming for holds for φ and φ'.

Claim 2. *There is a satisfying assignment for φ if and only if there exists a satisfying assignment for φ'.*

Proof. Let's consider both directions:

$\Longleftarrow$ If there is an assignment that satisfies all of the clauses of φ', then C'_{2i-1} and C'_{2i} should be both satisfied. Because z and $\overline{z}$ take on different values, it must be the case that one of C'_{2i-1} or C'_{2i} is satisfied by its other literals, which are also in the original clause C_i. From our construction, it then follows that C_i is satisfied by the same assignment, ignoring the extra variables. As this holds for all clauses, there is a satisfying assignment for φ.

$\Longrightarrow$ Assume there is a satisfying assignment for φ. Since clause C_i is satisfied, it must be the case that $x_p = 1$ or $y_q = 0$ for $1 \leq p \leq s$ and $1 \leq q \leq t$. If some $x_p = 1$, we assign 0 to z, making $\overline{z}$ true, otherwise we assign 1 to z. In this case, both C'_{2i-1} and C'_{2i} will be true. As this holds for every i, this yields a satisfying assignment for φ'.

□

Note that φ' contains at most $n+m$ variables and $2m$ clauses. Moreover, each of its clauses can be computed in polynomial time from φ. It follows that the reduction is computable in polynomial time and that MONO-SAT is NP-hard.

Problem 2

Consider *not-all-equal SAT* (NAE-SAT): Given a CNF formula, decide if there exists an assignment such that each clause contains at least one true literal and one false literal.

(a) Show that NAE-4-SAT is NP-hard using a reduction from 3-SAT.

(b) Show that NAE-3-SAT is NP-hard using a reduction from NAE-4-SAT.

Part (a)

Let φ be an instance for 3-SAT with m clauses and n variables. Fix some fresh variable z, and for each clause $(\ell_i \vee \ell_j \vee \ell_k)$ in φ, we add the clause $(\ell_i \vee \ell_j \vee \ell_k \vee z)$ to φ'.

Claim 3. *φ is satisfiable if and only if there exists a not-all-equal satisfying assignment for φ'.*

Proof. Let's consider both directions:

$\implies$ Suppose X is a satisfying assignment for φ, let's consider the assignment

$$X' = X \cup \{z = 0\}$$

for φ'. Each clause in φ' contains at least one true literal under X' since X assigns at least one literal in each clause of φ to 1. By setting $z = 0$ we ensure that each clause in φ' contains at least one negative literal. Thus $X' = X \cup \{z = 0\}$ is a not-all-equal assignment for φ'.

$\impliedby$ Suppose that there exists a not-all-equal satisfying assignment X' for φ'. We have two cases.

- If $z = 0$ in X', then the assignment $X = X' \setminus \{z = 0\}$ will set at least one literal in each clause in φ' to 1, and therefore at least one literal in each clause in φ to 1. Hence X is a satisfying assignment for φ.
- If $z = 1$, then the assignment $X = X' \setminus \{z = 1\}$ will assign at least one literal in each clause of φ' to 0. That is, at least one literal in each clause of φ is assigned 0. Therefore, the assignment $\overline{X}$, by negating all assignments in X, assigns at least one literal in each clause of φ to 1. Hence $\overline{X}$ is a satisfying assignment for φ.

Hence in either case, we can find a not-all-equal assignment for φ.

□

By construction, φ' contains m clauses and $n + 1$ variables, and each clause in φ' contains at most 4 literals. Hence φ' is a valid instance for NAE-4-SAT and can be constructed in polynomial time.

Because 3-SAT can be reduced to NAE-4-SAT in polynomial time and 3-SAT is NP-hard, NAE-4-SAT is also NP-hard.

Part (b)

Let φ be an instance for the NAE-4-SAT problem with m clauses and n variables. We construct an instance for NAE-3-SAT as follows: For each clause $(\ell_i \vee \ell_j \vee \ell_k \vee \ell_l)$ in φ, add two clauses $(\ell_i \vee \ell_j \vee z)$ and $(\overline{z} \vee \ell_k \vee \ell_l)$ to φ', for some fresh variable z.

Claim 4. *There exists a not-all-equal satisfying assignment for φ if and only if there exists a not-all-equal satisfying assignment for φ'.*

Proof. Let's consider both directions:

$\Longleftarrow$ Let X' be a not-all-equal assignment for φ'. Then for the two clauses $(\ell_i \vee \ell_j \vee z)$ and $(\overline{z} \vee \ell_k \vee \ell_l)$, if z is assigned 1, then there exists at least one literal between ℓ_i and ℓ_j assigned to 0, and there exists at least one literal between ℓ_k and ℓ_l assigned to 1. Thus there will be at least one true literal and at least one false literal in each clause $(\ell_i \vee \ell_j \vee \ell_k \vee \ell_l)$, and thus $X' \setminus \{z = 1\}$ is a not-all-equal assignment for φ. The argument follows similarly if z is assigned 0.

$\Longrightarrow$ Suppose that there exists a not-all-equal assignment X for φ. Then in each clause $(\ell_i \vee \ell_j \vee \ell_k \vee \ell_l)$, there exists at least one true literal and at least one false literal. Consider two groups: (1) ℓ_i and ℓ_j, and (2) ℓ_k and ℓ_l.

- If both groups contain one true literal and one false literals, then $(\ell_i \vee \ell_j \vee z)$ and $(\overline{z} \vee \ell_k \vee \ell_l)$ both contain at least one true literal and false literal regardless the assignment of z.
- If any group contains all true literals (or all false literals) under X, then the other group would contain at least one false literal (or true literal), and thus we could set $z = 0$ (or $z = 1$), ensuring that $(\ell_i \vee \ell_j \vee z)$ and $(\overline{z} \vee \ell_k \vee \ell_l)$ both contain at least one true literal and one false literal.

In either case, we can find a not-all-equal satisfying assignment for φ.

□

By construction, each clause in φ' contains at most 3 literals, and φ' contains at most $m + n$ variables and $2m$ clauses. Thus φ' can be constructed in polynomial time.

Because NAE-4-SAT can be reduced to NAE-3-SAT in polynomial time and NAE-4-SAT is NP-hard, NAE-3-SAT is also NP-hard.

Problem 3

Show that the following problem is NP-complete: Given two lists of games (one per conference), and a list of developers for each game, decide whether it is possible for each conference to have every game on its list be presented by one of its developers such that no developer needs to attend both conferences.

To prove that the given problem is NP-complete, we show that it is both in NP and NP-hard. We first establish that is in NP: consider an instance for the problem, defined by two lists of games (one for the conference in Beijing and one for the conference in Chicago) and a list of developers per game. Consider also a potential solution the problem, which is an assignment of developers to games/conferences (since the same game may be showcased on both conferences). To verify whether the solution is valid or not, it suffices to check whether there are no developers holding presentations at both conferences, and that every game being showcased in a conference is presented by at least one developer. This can be done in polynomial time on the number of games and developers, so we conclude that the given problem is in NP.

To prove that the given problem is NP-hard, we present a reduction from MONO-SAT. The idea is to create one developer per variable and one game per clause, which has as developers the developers representing the literals in the original clause. If a clause only contains non-negated literals, we showcase the corresponding game in Beijing, and if a clause only contains negated literals, we showcase the corresponding game in Chicago. Finally, choosing a developer to showcase a game at a conference is equivalent to setting its corresponding literal to true. The constraint that a developer can present at only one conference guarantees that we don't set both a variable and its negation to true, and the constraint that all games must be presented guarantees that we satisfy all clauses in the original formula. We now present this idea more formally.

Let φ be an instance of MONO-SAT with n variables and m clauses. For each clause containing only positive literals

$$C_i = x_1 \vee x_2 \vee \cdots \vee x_i,$$

create a game G_i with developers $\{x_1, x_2, \ldots, x_i\}$, and showcase it in Beijing. For each clause containing only negative literals

$$C_i = \overline{y_1} \vee \overline{y_2} \vee \ldots \overline{y_j},$$

create a game G_i with developers $\{y_1, y_2, \ldots, y_j\}$, and showcase it in Chicago. This construction introduces m games and n developers, and can be computed in polynomial time.

Now, it suffices to establish the following claim.

Claim 5. *The formula φ is satisfiable if and only if it is possible for each conference to have every game on its list be presented by one of its developers such that no developer needs to attend both conferences.*

Proof. Let us consider both directions.

$\Longrightarrow$ Assume that φ is satisfiable and let X be a satisfying assignment for it. Consider a clause C_i with only non-negated variables. Since this clause needs to be satisfied by X, there must be a variable x in C_i that is set to true by X. Pick developer x to present game G_i in Beijing. Now consider a clause C_i with only negated variables. Again, this clause needs to be satisfied by X, so there must be a variable x in C_i set to false by X. Pick developer x to present game

G_j in Chicago. By construction, it follows that every game for each conference is presented by some developer. Moreover, because a variable can only be set to either true or false, we are guaranteed that no developer needs to attend both conferences.

$\Longleftarrow$ Assume that it is possible for each conference to have every game on its list be presented by one of its developers such that no developer needs to attend both conferences. Fix one possible assignment of developers to games/conferences. If a developer is scheduled to present a game in Beijing, set its corresponding variable to true, and if they are scheduled to present a game in Chicago, set their corresponding variable to false. If there are any variables not set after this, set those to true. This produces a valid assignment because no developer needs to attend both conferences, so we will never set a variable to both true and false. Moreover, as each conference has every game in its list being presented by some developer, this assignment satisfies all clauses of φ, and therefore satisfies φ itself.

□

Since MONO-SAT is NP-hard, and the reduction from MONO-SAT to the given problem runs in polynomial time, we conclude that the given problem is also NP-hard.

Problem 4

Show that the following problem is NP-hard: Given a 3-CNF formula, does there exist an assignment that satisfies exactly one literal of every clause?

We argue that the problem is NP-hard by presenting a polynomial-time mapping reduction from 3-coloring to it. The intuition for using 3-coloring is that we can model choosing one of the three colors for a vertex as choosing one of three variables of a clause. (In addition, we will need to enforce the condition that for each edge its two vertices have different colors.)

An instance of 3-coloring consists of a graph $G = (V, E)$. It is a yes-instance if it is possible to assign colors to the vertices such that for all $(v_1, v_2) \in E$, v_1 is a different color than v_2. We express the 3-colorability of G as an instance φ of the problem at hand such that the valid 3-colorings of G are in 1-to-1 and onto correspondence with the valid assignments to φ.

In order to do so, we introduce a Boolean variable $c_{i,j}$ for each $1 \leq i \leq 3$ and $1 \leq j \leq |V|$. The intended meaning of $c_{i,j}$ is to indicate whether vertex j is colored with color i. To enforce that the assignment corresponds to a valid coloring, we include two types of clauses:

- Vertex clauses, which enforce that every vertex receives exactly one of the three colors. The clause for the j-th vertex is $(c_{1,j} \vee c_{2,j} \vee c_{3,j})$.
- Edge clauses, which enforce that the coloring is valid. For each color i and edge $(j, k) \in E$ we add a clause $(c_{i,j} \vee c_{i,k} \vee d_{i,j,k})$. Here $d_{i,j,k}$ is a variable which is not used in any other clause in the formula. Since we must choose exactly one of these three variables, if one of the two vertices is colored with i then the other one cannot be colored with i. If neither of them are colored i then we can set $d_{i,j,k} = \mathit{true}$ to satisfy this clause.

Claim 6. *The formula constructed above has an assignment that satisfies exactly one literal of each clause iff G can be colored with three colors.*

Proof. Assume there is such an assignment. Then color each vertex j with color i if $c_{i,j} = \mathit{true}$ in the assignment. We only use three colors, because i only takes three values. Each vertex is colored with exactly one color, because the clause representing that vertex has exactly one satisfied variable. Moreover for every edge $(j, k) \in E$, j and k do not have the same color, otherwise $(c_{i,j} \vee c_{i,k} \vee d_{i,j,k})$, which is a clause of the formula, would have more than one variable satisfied. So the coloring is valid.

Assume there is a valid 3-coloring of G. For each vertex j which is colored with color i, set $c_{i,j} = \mathit{true}$. For every edge $(j, k) \in E$ and every color i where neither j or k are colored with i, set $d_{i,j,k} = \mathit{true}$. Set every other variable to false. This is an assignment which satisfies exactly one variable of every clause. Each vertex has exactly one color, which takes care of the clauses corresponding to vertices. For each edge (j, k) and each color i, either one of j or k is colored with i, or neither is. In the first case $c_{i,j} = \mathit{true}$ or $c_{i,k} = \mathit{true}$ (but not both), and in the second case $d_{i,j,k} = \mathit{true}$. So the clause $(c_{i,j} \vee c_{i,k} \vee d_{i,j,k})$ is taken care of. □

There is one clause for each vertex and one clause for each edge, and we can construct the formula in $O(|V| + |E|)$ time. Thus, we have established a polynomial-time mapping reduction from the NP-hard problem 3-coloring to the problem at hand, which shows that the problem at hand is NP-hard.

Problem 5

You are given a list of formulas of the following form:

- $x_i = 0$,
- $x_i = 1$,
- $x_i \geq x_j$ or $x_k < X$, and
- $x_i > x_j$ or $x_k \leq X$.

Here X denotes a set of variables, and $x_k < X$ means that $x_k < x$ for all $x \in X$; $x_k \leq X$ is defined similarly. The question is whether there exists a way to assign the values 0 and 1 to the variables such that all formulas are satisfied.

Design a polynomial-time mapping reduction from 3-SAT to this problem.

Let us denote the problem at hand by PL (for Programming Language problem). We show a polynomial-time mapping reduction from the decision problem 3-SAT to PL.

Consider an arbitrary 3-CNF clause, say

$$l_1 \vee l_2 \vee l_3 \tag{1}$$

where each l_i is a literal. The 3-CNF clause (1) is logically equivalent to

$$l_1 \geq \overline{l_2} \text{ or } 0 < \{l_3\}. \tag{2}$$

Expression (2) is almost of the third allowed type except for the negation and the use of the constant 0. The latter problem we can remedy by replacing 0 by a variable y_N which is defined to be 0 by a formula of the first type:

$$y_N = 0. \tag{3}$$

That is, we can rewrite (2) as

$$l_1 \geq \overline{l_2} \text{ or } y_N < \{l_3\}. \tag{4}$$

In order to deal with the negation, we can introduce PL variables for all 3-CNF literals , i.e., for every 3-CNF variable z_i , we introduce a PL variable x_i (intended to replace z_i), and another PL variable y_i (intended to replace $\overline{z_i}$).

What remains is to express that x_i is the complement of y_i in the PL formalism. We claim that

$$(x_i = \overline{y_i}) \Longleftrightarrow ((x_i < y_i) \vee (y_i \leq 0)) \wedge ((y_i < x_i) \vee (y_i \geq 1)). \tag{5}$$

The truth table below proves the validity of our claim.

x_i	y_i	$(x_i < y_i) \vee (y_i \leq 0)$	$(y_i < x_i) \vee (y_i \geq 1)$
0	0	T	F
0	1	T	T
1	0	T	T
1	1	F	T

Using one additional variable, x_P, satisfying

$$x_P = 1, \tag{6}$$

we can cast the right-hand side of (5) as two PL formulas of type 4, namely

$$y_i > x_i \text{ or } y_i \leq \{y_N\} \tag{7}$$

$$x_i > y_i \text{ or } x_P \leq \{y_i\}. \tag{8}$$

Now, we are ready to define a polynomial-time mapping reduction f from 3-SAT to PL. Let φ be an arbitrary 3-CNF formula. First, by repeating literals as needed, we can transform any clause in φ with less than 3 literals into an equivalent 3-CNF clause with exactly three literals. For example, we can replace $x_1 \vee \overline{x_2}$ by $x_1 \vee \overline{x_2} \vee \overline{x_2}$.

We define $f(\varphi)$ to consist of the following formulas:

- the formulas (3) and (6),
- the formulas (7) and (8) for every variable z_i in φ, and
- the formula (4) for every clause of the form (1) in φ where we replace l_1, l_2 and l_3 in (4) by the appropriate PL variables x_i or y_i .

The translation f can be computed in polynomial time, and by construction, $\varphi \in$ 3-SAT iff $f(\varphi) \in$ PL.