CS 577: Introduction to Algorithms Fall 2022

Homework 10

Instructor: Dieter van Melkebeek TA: Nicollas Mocelin Sdroievski

This homework covers NP-completeness of Satisfiability and closely related problems. **Problem 3 must be submitted for grading by 2:29pm on 12/6.** Please refer to the homework guidelines on Canvas for detailed instructions.

Warm-up problems

1. A CNF formula is *monotone* if the literals in each clause are either all positive or all negative. For example, if we have

   - $\varphi_1 = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_1 \vee x_3 \vee x_4)$ and
   - $\varphi_2 = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_3 \vee x_4)$,

   then φ_1 is monotone but φ_2 is not.

   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.

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. In NAE-k-SAT, each clause has at most k literals.

   For example, consider

   - $\varphi_1 = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_3} \vee x_1)$ and
   - $\varphi_2 = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_3 \vee x_4)$.

   Both φ_1 and φ_2 have a satisfying assignment, but φ_1 is a NO instance for NAE-SAT since its only satisfying assignment assigns all literals in the clause $(x_1 \vee x_2 \vee x_3)$ to 1. φ_2 is a YES instance for NAE-SAT since we can assign $x_1 = 1$, $x_2 = 1$, $x_3 = 0$ and $x_4 = 1$.

   (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.

Regular problems

3. [Graded] Due to unfortunate planning, two game development conferences are happening simultaneously in Beijing and Chicago. Each conference is showcasing several games, and each game is supposed to be presented in person by one of its developers. Developers often work on multiple games, and may present on any subset of the games they have worked on. However, it is infeasible for a single developer to attend both conferences in person. The conferences' organizers would like to know whether it is possible to assign presentations to developers so that no developer has to present at both conferences.

   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.

   The following are examples of instances for the problem and their respective answers.

   Example 1. The conference in Beijing is showcasing games G_1, G_2 and G_3, and the conference in Chicago is showcasing games G_1 and G_4. The following is a list of developers for each game:

   - G_1: $\{d_1, d_2\}$.
   - G_2: $\{d_1, d_3\}$.
   - G_3: $\{d_4\}$.
   - G_4: $\{d_3, d_4\}$.

   In this example, having d_1 present games G_1 and G_2, and d_4 present G_3 in Beijing, while having d_2 present G_1 and d_3 present G_4 in Chicago is a valid solution, since no developer needs to present at both conferences. In this case the answer would be yes. Note that it is not a problem for d_1 to present games G_1 and G_2 in the same conference. Note also that is is allowed for the two conferences to showcase the same game.

   Example 2. The conference in Beijing is showcasing games G_1, G_2 and G_3, and the conference in Chicago is showcasing games G_4 and G_5. The following is a list of developers for each game:

   - G_1: $\{d_1, d_3\}$.
   - G_2: $\{d_2, d_3\}$.
   - G_3: $\{d_1, d_2, d_3\}$.
   - G_4: $\{d_3\}$.
   - G_5: $\{d_1, d_2\}$.

   In this example, it does not matter how we pick developers to present games, we always end up with some developer needing to present at both conferences. Note that we must pick d_3 to present G_4. After that, the only choice for G_2 is d_2, who can also present G_3. We are then left with G_1 in Beijing and G_5 in Chicago, but only a single developer to present both. Therefore, the answer is no.

4. In class we developed a polynomial-time algorithm for finding a satisfying assignment to a given 2-CNF formula (or report that none exists). Show that the following variant is NP-hard: Given a satisfiable 2-CNF formula, find a satisfying assignment that sets the smallest number of variables to true.

5. Some time ago, people from the programming languages group asked about the following problem.

   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.

Challenge problem

6. Given a Boolean circuit C, you want to find a Boolean circuit C' that behaves the same as C on every input but has as few gates as possible. Show that if P = NP, then this problem can be solved in polynomial time.

Programming problem

7. SPOJ problem TWOSAT (https://vn.spoj.com/problems/TWOSAT/). As this problem is only available on the Vietnamese SPOJ website, an English translation is provided on the following pages.

TWOSAT - Travel

A travel company is organizing a group of foreign tourists to travel to M cities in Vietnam. Each visitor has two requirements of the form "Want to go to c" or "Do not want to go to c", for some city c. These tourists are very difficult, and want at least one of their requirements to be met. The travel agency is struggling to choose a set of cities to visit that appeases all the tourists. You are tasked with helping them solve this problem.

Input Line 1 contains two integers, N and M ($1 \leq N \leq 20000$, $1 \leq M \leq 8000$). They are the number of tourists and cities, respectively. The cities are numbered 1 through M.

Each of the next N lines contains two integers u, v with $1 \leq |u| \leq M$ and $1 \leq |v| \leq M$, encoding the request of one of the tourists. A positive integer indicates the customer wants to go to that city, while a negative integer indicates the customer does not want to go to that city.

Output On line 1, write `YES` if there is a plan that satisfies at least one requirement of every visitor; write `NO` in the opposite case.

If there is a plan, the next two lines of output shall encode one such plan. On line 2, write a positive integer k, indicating how many cities will be visited in the plan. On line 3, write k integers, the indices of the cities to visit.

Example 1

Input:

```
2 3
-1 -2
1 2
```

Output:

```
YES
2
2 3
```

Example 2

Input:

```
3 3
-1 2
-1 -2
1 -2
```

Output:

```
YES
0
```

Example 3

Input:

```
4 3
-1 2
-1 -2
1 -2
1 2
```

Output:

```
NO
```


Homework 10 Solutions to Warm-up Problems

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 $C_i = x_1 \vee x_2 \vee \cdots \vee x_s \vee \overline{y_1} \vee \overline{y_2} \vee \dots \overline{y_t}$ in φ, add two clauses $C'_{2i-1} = x_1 \vee x_2 \vee \cdots \vee x_s \vee z$ and $C'_{2i} = \overline{z} \vee \overline{y_1} \vee \overline{y_2} \vee \dots \overline{y_t}$ 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}:

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

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. The basic idea is as follows: If we add a variable to each clause in φ (obtaining clauses with 4 variables), then we should be able to set it to false when the original clause is satisfiable, guaranteeing that there is at least one variable per clause that is set to false. Of course, we should make sure that the formulas are equivalent, in the sense that the original formula should be satisfiable if and only the new formula with one extra variable is not-all-equal satisfiable, but we can use the not-all-equal property to make sure that this is the case. We now present this idea in more detail. 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:

$\Longrightarrow$ 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'.

$\Longleftarrow$ 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.

CS 577: Introduction to Algorithms	Fall 2022
Homework 10 Solutions to Regular Problems	
Instructor: Dieter van Melkebeek	TA: Nicollas Mocelin Sdroievski

Problem 3

> Show that the following problem is NP-hard: 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 to the problem, which is an assignment of developers to games/conferences (since the same game may be showcased at 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 linear, and therefore polynomial, time.

Now, it suffices to establish the following claim.

Claim 1. *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

> In class we developed a polynomial-time algorithm for finding a satisfying assignment to a given 2-CNF formula (or report that none exists). Show that the following variant is NP-hard: Given a satisfiable 2-CNF formula, find a satisfying assignment that sets the smallest number of variables to true.

First, we note that there is a natural equivalence between this version of the problem and the version where we wish to find a satisfying assignment that sets the largest number of variables to true: just replace every occurrence of a literal x_i by $\overline{x_i}$ and $\overline{x_i}$ by x_i. With that in mind, it suffices to prove that this variant is NP-hard, which we do by reducing from Independent Set.

Recall the reduction from Independent Set to CNF-SAT as seen in class. On input (G, k), it introduces one variable x_v for each vertex v in G, which indicates whether v belongs to an independent set. It also introduces the clause $(\overline{x_u} \vee \overline{x_v})$ for each edge $e = (u, v)$, which enforces the independence condition. Notice that, up until this point, the reduction has constructed a 2-CNF-SAT instance. The rest of the reduction introduces variables and clauses that enforce the size requirement. In our case, however, those are unnecessary, since the problem we are reducing to already sets the largest number of variables to true (meaning it finds the largest independent set).

Let us formalize the reduction. On input (G, k) with n vertices and m edges, create a formula φ as follows: Introduce a variable x_v for each vertex v in G and introduce a clause $(\overline{x_u} \vee \overline{x_v})$ for each edge $e = (u, v)$. Call the black-box with input φ, obtaining a satisfying assignment X. Finally, output yes if and only the number of variables set to true by X is greater than or equal to k.

We now argue correctness. First, note that the formula φ constructed by the reduction is always satisfiable (just set every variable to false). In that case, correctness amounts to arguing the following claim.

Claim 2. *G has an independent set of size k if and only if φ has a satisfying assignment that sets k variables to true.*

Proof. $\Longrightarrow$ Assume that G has an independent set I of size k. Construct an assignment X for φ as follows: For each $v \in I$ set x_v to true, and set the remaining variables to false. This assignment sets exactly k variables to true, so it remains to check that it satisfies φ. It is easy to see that X satisfies every clause that does not involve a variable x_v for $v \in I$. Now, consider some clause $(\overline{x_u} \vee \overline{x_v})$ for $v \in I$. Because the edge (u, v) is in G and I is an independent set, it must be the case that $u \notin V$, meaning x_u is set to false and the clause is satisfied.

$\Longleftarrow$ Assume that φ has a satisfying assignment X that sets k variables to true, and let I be the set of vertices corresponding to the variables set to true by X. Clearly, $|I| = k$, so it remains to check that it is an independent set. Consider an edge (u, v) of G such that $v \in I$. Because it is the case that X sets x_v to true and satisfies the clause $(\overline{x_u} \vee \overline{x_v})$, it must be the case that x_u is set to false, meaning it is not in I.

□

As for the running time, φ has n variables and m clauses (each with two variables), and can be constructed time linear in n and m. We conclude that the reduction runs in polynomial time and that the given problem is NP-hard.

Part (b)

Let φ be an instance for the NAE-4-SAT problem with m clauses and n variables. For each of φ's clauses, our approach is to construct two NAE-3-SAT clauses that together are equivalent to the original clause, in the sense that an assignment that not-all-equal satisfies the original clause can be transformed into one that not-all-equal satisfies the new clauses, and vice versa. This can be done by splitting the four literals in the original clause into two clauses, with an extra variable that "ties" both clauses together. More formally, 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 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

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

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.

COMP SCI 577 Homework 10 Problem 3

Computational Intractability

Ruixuan Tu

rtu7@wisc.edu

University of Wisconsin-Madison

5 December 2022

Algorithm

For any MONO-SAT CNF formula φ, our approach is to convert every clause inside the CNF formula to one distinct game $G_i := \{d_{i_1}, d_{i_2}, \dots\}$. If the clause is in the form $d_{i_1} \vee d_{i_2} \vee \cdots$, then we put the game on the list for Beijing. Otherwise, if the clause is in the form $\neg d_{i_1} \vee \neg d_{i_2} \vee \neg \cdots$, then we put the game on the list for Chicago. In this way, no game should be presented in both Chicago and Beijing, but some developers could choose whether to present for the set of related games in Chicago or another distinct set of related games in Beijing. We use d_{i_j} to reindex the developers $d_1, d_2, \dots$, keeping only that needed to be in the set for every specific clause. Denote φ' to be the problem to determine 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. Then we solve the MONO-SAT CNF satisfiability problem determining φ by the conference planning problem determining φ'.

Proof

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

Proof. We consider both directions

$\Rightarrow$ Assume there is a satisfying assignment X for φ. As every d_i could be only assigned to one value, it matches the condition that a developer could only be in either Beijing (when assigned true) or Chicago (when assigned false). For every clause $d_{i_1} \vee d_{i_2} \vee \cdots$ assigned by X, then there is a corresponding game G_i in Beijing, i.e., $\exists d_{i_{j'}} \in [d_{i_j}]$ which will present in Beijing ($d_{i_{j'}}$ is

true $\iff$ $\neg d_{i_{j'}}$ is false $\iff$ $d_{i_{j'}}$ is in Beijing but not Chicago), so that the game must be able to be presented in Beijing. For every clause $\neg d_{i_1} \vee \neg d_{i_2} \vee \neg \cdots$ assigned by X, then there is a corresponding game G_i in Chicago, i.e., $\exists d_{i_{j'}} \in [d_{i_j}]$ which will present in Chicago ($\neg d_{i_{j'}}$ is true $\iff$ $d_{i_{j'}}$ is false $\iff$ $d_{i_{j'}}$ is in Chicago but not Beijing), so that the game must be able to be presented in Chicago. Thus, with assignment X, all the games on the two lists must be presented by at least one developer for each game, so that φ' is satisfied. □

$\Leftarrow$ Assume there is a satisfying assignment X' for φ'. For every game G_i which could be presented in Beijing, there is at least one developer $\exists d_{i_{j'}} \in [d_{i_j}]$, i.e., $d_{i_{j'}}$ is true, so that the clause $d_{i_1} \vee d_{i_2} \vee \cdots$ is true in φ. For every game G_i which could be presented in Chicago, there is at least one developer $\exists d_{i_{j'}} \in [d_{i_j}]$, i.e., $d_{i_{j'}}$ is false, so that the clause $\neg d_{i_1} \vee \neg d_{i_2} \vee \neg \cdots$ is true in φ. Thus, φ is satisfied. ■

Claim. MONO-SAT $\leq^P$ Conference Planning

Proof. The algorithm must terminate as there is no recursive call. In this algorithm, we only do one loop to iterate over all clauses once, determine if it is all positive or negative ($\neg$), put all sub-elements as d_{i_j} into G_i, and assign G_i to either Beijing or Chicago according to the sign. Thus we construct a problem in Conference Planning from a problem in MONO-SAT using $O(n)$ in polynomial time. ■

To wrap up, as We have already proved that MONO-SAT is NP-hard in Problem 01 of the warmup problems of this assignment, Conference Planning is NP-hard. ■