15-455: UCT **K. Sutner**

Assignment 4 Due: **February 15**, 2024, 24:00.

1. Minimization Problems (30)

Background

A minimization problem consists of $\mathfrak{M} = \langle I, \mathsf{sol}, \mathsf{cost}\rangle$ where $I \subseteq \Sigma^\star$ is a collection of instances, $\mathsf{sol} : I \to \mathfrak{P}_{\mathsf{fin}}(\Sigma^\star)$ (codomain is all finite subsets of $\Sigma^\star$) is a solution function, and $\mathsf{cost} : \bigcup_{x\in I} \mathsf{sol}(x) \to \mathbb{N}$ is a cost function. Quite a few combinatorial problems can be interpreted as minimization problems. As an example, take Vertex Cover: here I is the set of all undirected graphs $\mathsf{sol}(x)$ is the collection of all vertex covers of graph x, and $\mathsf{cost}(w)$ is the cardinality of the vertex cover w. Any reasonable coding convention for I and $\mathsf{sol}(x)$ is fine.

We can associate three variants with $\mathfrak{M}$: the decision version $\mathfrak{M}_D$, the "counting" version $\mathfrak{M}_C$, and the search version $\mathfrak{M}_S$. We will ignore the function version here, since it requires some additional ordering on the solutions so we can talk about the "first solution."

Problem:	$\mathfrak{M}_D$
Instance:	$x \in I$, $k \in \mathbb{N}$.
Question:	Is there a $w \in \mathsf{sol}(x)$ such that $\mathsf{cost}(w) \leq k$?

Problem:	$\mathfrak{M}_C$
Instance:	$x \in I$.
Solution:	Determine the minimal cost of any $w \in \mathsf{sol}(x)$.

Problem:	$\mathfrak{M}_S$
Instance:	$x \in I$.
Question:	Determine a $w \in \mathsf{sol}(x)$ of minimal cost.

It is understood that an algorithm for $\mathfrak{M}_C$ or $\mathfrak{M}_S$ returns "No" if the solution set of instance x is empty.

Task

A. What does all this mean in the case of Independent Set and Clique?

B. Show that for Independent Set and Clique all three versions are polynomial time Turing equivalent.

C. Come up with one more example of a natural minimization problem.

D. We would like $\mathfrak{M}_D$, $\mathfrak{M}_C$ and $\mathfrak{M}_S$ to be polynomial time Turing equivalent (as in part (B)). What conditions on $\mathfrak{M}$ are needed to make this equivalence work?

Comment

Make sure your answer to part (D) covers standard examples like Vertex Cover, Independent Set and Clique. Try to be a bit more general, but don't worry about finding the most general conditions possible, it's fairly messy.

2. More Satisfiability (30)

Background

Here is a yet another variant of satisfiability: Unequal-3-Satisfiability (UE3SAT) where an instance is a formula in 3-CNF and we are looking for a satisfying truth assignment that makes at least one literal in each clause false. This may sound a bit strange, but it is often useful in reductions.

Recall from a previous HW that 2SAT is solvable in polynomial time. Here is a variant that is hard: given a 2-CNF formula and a bound k, determine whether there is a truth assignment that satisfies at least k of the clauses. So the special case when k is the number of clauses is easy. Let's refer to this version as Counting 2SAT (CNT2SAT).

Task

A. Show that UE3SAT is $\mathbb{NP}$-complete.

B. Show that CNT2SAT is $\mathbb{NP}$-complete.

Comment

Both parts can be handled by a reduction from 3SAT, but you might be better off starting with UE3SAT for part (B).

3. Tilings (40)

Background

Informally, a Tiling Problem consists of a collection of square tiles with colored edges. We want to know whether it is possible to place the tiles on an infinite chess board in a way that the colors of adjacent edges match, filling up the whole board in the process. The tiles cannot be rotated or reflected, only translated; we assume an unlimited supply of tiles of each type. This infinite version of the problem is undecidable; the proof rests on encodings of a computation of a Turing machine and is quite messy (the original proof used a tile set of cardinality 20,426).

To push things down to $\mathbb{NP}$, we restrict our tilings to a board of size $n \times n$. Technically, we have a finite set C of colors and a tile set $T \subseteq C^4$. A tiling is now a placement of n^2 tiles so that adjacent tiles share the same color. In the anchored square tiling problem (ASTP) we are given an instance $\langle C, T, 0^n, t_1, t_2 \rangle$: the colors, the tiles, the grid size n (coded in unary as 0^n) and two anchor tiles. Tile t_1 must be placed in the North-West corner of the grid, and t_2 in the South-West corner.

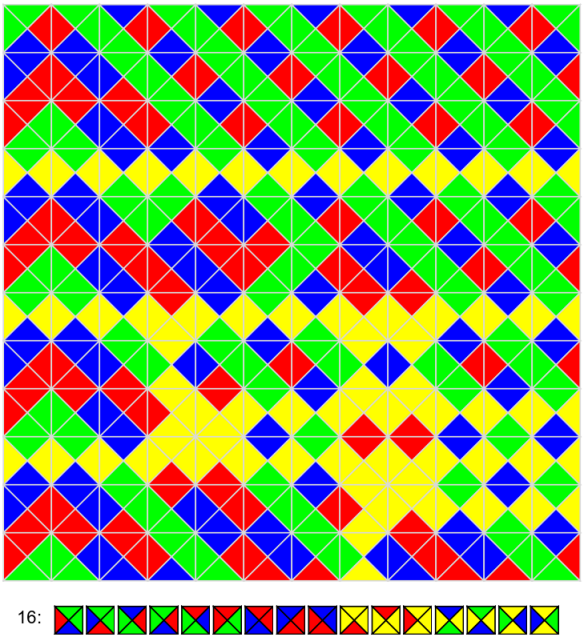
Task

A. Show that ASTP is in $\mathbb{NP}$.

B. Show that ASTP is $\mathbb{NP}$-hard by a direct simulation of polynomial time Turing machines.

C. Explain how to get rid of the South-West anchor tile, without affecting hardness.

Comment Here is an example of a 12×12 tiling using 16 tiles with colors red, green, blue and yellow.



Comment For the simulation in part (B), think of row k in the tiling as corresponding to the configuration of the machine at time k. This won't quite work out, you will need several rows to simulate a single step of the Turing machine. Also, you may want to make some harmless assumptions about the machine to simplify the construction.