Combinatorial Game Complexity

Swapnil Garg

December 2016

1 Introduction

Computational complexity theory is a computational theory that classifies problems according to how complex an algorithm needs to be to solve them. A computational problem is a problem that specifies an input, formalized as a string over an alphabet, and requires an output. The complexity of an algorithm is viewed as how complex in time or space an algorithm becomes in relation to the length of the input, rather than a specific instance of the input. For example, given a board in the game Go, whether the first player will win or lose can be computed in constant time. However, the game Go considered as any NxN board does not have a constant time solution, as the complexity increases with the size of the board.
These problems are often phrased as decision problems, which output either "yes" or "no." For example, the problem of factorizing a number N can be phrased as whether or not N has a prime factor less than a number k. Below is a table of common complexity classes.

Class	Constraint
L	logarithmic space
P	polynomial time
NP	given a proof, verifiable in polynomial time
PSPACE	polynomial space
EXPTIME	exponential time

It is known that L $\subset$ P $\subset$ NP $\subset$ PSPACE $\subset$ EXPTIME. It is also known that L is strictly contained in PSPACE and P is strictly contained in EXPTIME, but it is unknown which relations above are strict. Specifically, the P=NP problem asks whether or not P=NP.

2 NP-Completeness

For a class X, a problem L is X-hard if it is at least as hard as all problems in X: all problems in X can be reduced to polynomial time plus a polynomial number of calls to L. A problem L is X-complete if it is both X and X-hard.

One of the first problems proven to be NP-complete was SAT, or satisfiability, given by the Cook-Levin Theorem. Given a boolean expression, it asks whether there is a value for each variable (true and false) that makes the whole expression true. Many other problems reduce to this problem, such as the ones below.

Definition. Given a graph G and a subset of vertices S with edge weights, the Steiner Tree problem asks what the total weight is of a spanning tree of G (a subset of edges of G which forms a tree) that includes all of S. Phrased as a decision problem, it asks whether there is a spanning tree spanning S with a total weight at most a value x.

Definition. Given a set $U = \{u_1, u_2, ..., u_k\}$ and a set $\{S_1, S_2, ..., S_l\}$ of subsets of U, the Exact Cover problem asks whether there is a set of S_i that are disjoint such that their union is U.

It turns out that both of the above problems are in NP.

Theorem 1. *The Steiner Tree problem is NP-complete.*

Proof. We use the following lemma without proof:

Lemma 2. *The exact cover problem is NP-complete.*

The proof is in "Reducibility Among Combinatorial Problems" by Karp (1972). It turns out that this reduces to satisfiability.

Now, for a set $U = \{u_1, u_2, ..., u_k\}$ and a set $S = \{S_1, S_2, ..., S_l\}$ where every S_i is a subset of some u_i. Let n_0 be a node, and consider the graph with nodes $n_0, S_1, S_2, ..., S_l, u_1, u_2, ..., u_k$. Suppose that there is an edge from n_0 to all S_i, with edge weight $|S_i|$, and there is an edge from S_i to u_j iff $u_j \in S_i$, with edge weight 0. If we ask whether there exists a spanning tree containing $n, u_1, u_2, ..., u_k$, this is equivalent to whether there is an exact cover over $\{u_1, u_2, ..., u_k\}$ using sets $\{S_1, S_2, ..., S_l\}$. □

In particular, note that this reduces to the subproblem of the Steiner Tree problem which asks to find the minimum spanning tree that spans one side of a bipartite graph, as $n_0 \cup U$ is one side and S is the other side.
For more about NP-Completeness, refer to Garey and Johnson's *Computers and Intractability: A Guide to the Theory of NP-Completeness* (1979).

3 Complexity of Games

Complexity can be applied to Combinatorial Game Theory by computing the value or outcome class of a position. Many games are EXPTIME-complete, such as generalized Chess, Go, and Checkers. Others, such as generalized Amazons and Tic-tac-toe, are PSPACE-complete. Given a one-player position of Amazons, computing the value of the position (the maximal number of moves left) is NP-hard.

Theorem 3. *Evaluating a Hackenbush position is NP-hard.*

Proof. Consider a piece of redwood furniture (a connected graph where all blue edges are connected to the ground and connected to a unique red edge called a leg) which we know has value $\frac{1}{2^n}$ for some n.
By the Don't-Break-It-Up Theorem, for a piece of redwood furniture G with value $\frac{1}{2^n}$, Right has a move to G' with value $\frac{1}{2^{n-1}}$ such that G' is still connected, called a worthwile move. Consider a redwood bed, a piece G of redwood furniture (a connected graph where all blue edges are connected to the ground and connected to a unique red edge called a leg) with the extra condition that all non-leg edges are connected to exactly one leg.

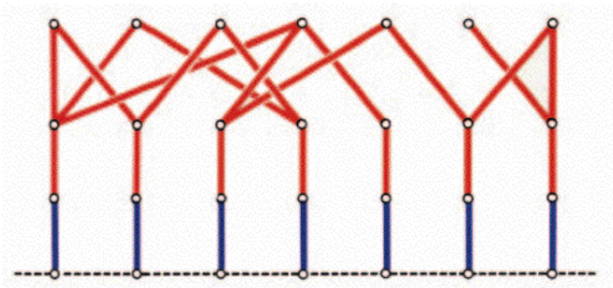

Figure 1: Redwood Bed (from *Winning Ways*)

By induction on the number of edges, we can prove that any redwood bed that is a tree has value $\frac{1}{2}$. Then, suppose that after performing m worthwile moves that don't disconnect the graph, there are no more moves left that don't disconnect the graph, so the value of G is $\frac{1}{2^{m+1}}$. However, any move we do from G to G^R has $G^R \geq 2 * G$, so if we make l moves without disconnecting the graph and l is the maximum such number of moves possible, we have $l \geq m$ and $G \leq \frac{1}{2^{l+1}}$, so $l = m$ and $g = \frac{1}{2^{l+1}}$.
Proofs of the cited results above are in *Winning Ways*, pages 211-217.
Therefore, finding the value of a redwood bed is equivalent to finding the minimum spanning tree of the red non-leg edges that still connects to every leg edge, which is equivalent to the bipartite Steiner Tree problem. □

As the standard NP-complete problem is satisfiability, the standard PSPACE-complete problem is the quantified boolean formula: a boolean expression preceded by a series of $\exists x_i$ and $\forall x_j$, for example $\exists x_1 \forall x_2 \exists x_3 \forall x_4 (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4)$, read as "there exists an x_1 such that all x_2 there exists an x_3 such that for all x_4 the expression $(x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4)$ is true, where $\vee$ is or and $\wedge$ is and.
This is equivalent to a two-person game where player one chooses x_1, then player two chooses x_2, and so on, and player one's goal is to make the expression true. Many games can be shown to be PSPACE-complete by reducing to QBF, such as Amazons.