

ACO Student Seminar: Some of my favorite open problems you may have never heard about

Richard Lipton

February 4, 2009

Suggested website for open problems: [The Open Problem Garden](#)

1 An Exponential Algorithm for Knapsack

Problem 1 (Knapsack Problem). Given $a_1, \dots, a_n, b \in \mathbb{Z}_+$, does there exist $x_1, \dots, x_n \in \{0, 1\}^n$ such that $\sum_{i=1}^n a_i x_i = b$?

This problem admits a $\tilde{O}(2^{\frac{n}{2}})$ algorithm. The basic idea is as follows: rewrite the expression as

$$\sum_{i=1}^{\frac{n}{2}} a_i x_i = b - \sum_{i=\frac{n}{2}+1}^n a_i x_i$$

and enumerate over the $2^{\frac{n}{2}}$ objects on each side to find a common element. This can be done in $\tilde{O}(2^{\frac{n}{2}})$ time, but also requires $2^{\frac{n}{2}}$ space.

Open Question. Does there exist an $\tilde{O}(2^{\frac{n}{3}})$ algorithm to solve the knapsack problem?

One possible approach is a generalization of the $\tilde{O}(2^{\frac{n}{2}})$ algorithm. For sets A and B let

$$A + B = \{a + b : a \in A, b \in B\}.$$

We are then asked to determine whether an element x belongs to $A + B$. It turns out this question can be answered in $\tilde{O}(|A| + |B|)$ time. This yields another open question:

Open Question. Given sets A , B , and C , and an element x . Does there exist an $\tilde{O}(|A| + |B| + |C|)$ algorithm to determine whether x belongs to $A + B + C$?

Applying the previous trick, this result would imply an $\tilde{O}(2^{\frac{n}{3}})$ time algorithm to solve the knapsack problem.

2 Complexity of the Square-root sum

Problem 2 (Square-root sum). Given $a_1, \dots, a_n; k$ determine whether $\sum_{i=1}^n \sqrt{a_i} \leq k$.

One may attempt to answer this question by looking at a high precision floating point approximation of the left and right hand sides, but this may require doubly exponential precision.

Open Question. What is the complexity of the square-root sum?

As a motivating example, consider the TSP. It is known that the TSP in the plane is \mathcal{NP} -hard, but it is not known whether this problem even belongs to \mathcal{NP} because of this roadblock.

One might look at the related question

$$\sum_{i=1}^n b_i \sqrt{a_i} \stackrel{?}{=} k$$

for inspiration. With the aid of randomization, this problem can be answered in polynomial time. Observe that

$$\alpha = \sum_{i=1}^n b_i \sqrt{a_i} - k$$

is an algebraic integer. In particular, if one of its conjugates is zero, then all of its conjugates are zero, and likewise, if one of its conjugates is non-zero, then all of its conjugates are non-zero. There exist 2^n conjugates and with reasonable probability, a random conjugate will return a zero because it is actually a zero and not because it is outside the computer's precision. Therefore, we randomly select a conjugate by randomizing the sign of each term in the above sum. Unfortunately, this nice idea does not extend to the inequality problem.

An idea to address this question is as follows. $x \geq y \Leftrightarrow x+r \geq y+r$ for all $r \in \mathbb{R}$. Perhaps some properly chosen r would allow us to shrink the problem. For example if $x, y, r \in \mathbb{Z}$ such that there exists a prime p such that $x+r = p \cdot \alpha$ and $y+r = p \cdot \beta$ with $\alpha < x$ and $\beta < y$, then $x \leq y \Leftrightarrow \alpha \leq \beta$, and so the problem has become smaller in some sense. Consider $x = 503$, $y = 1007$ and $r = 1$. Then $503 \geq 1007 \Leftrightarrow 504 \geq 1008 \Leftrightarrow 1 \geq 2$.

Although this may not produce an algorithm for the problem, to establish membership in \mathcal{NP} we would only need to demonstrate the existence of a proof of polynomial length.

3 Bait and Switch

Thus far, we have been unsuccessful in establishing lower bounds on the time complexity of a problem. With respect to the knapsack problem posed earlier, we have the following open question:

Open Question. Does there exist an instance of the knapsack problem that would require more than linear time?

More generally, we consider Boolean functions. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be some Boolean function. For inputs $x_1, \dots, x_n \in \{0, 1\}$, we can think of the computation of $f(x_1, \dots, x_n)$ in terms of a circuit. One way to establish a lower bound might be to define some metric on the gates in this circuit describing how far we are from f .

Open Question. For a circuit computing f , can we define a formal notion of progress at a given gate towards computing f ?

Part of the challenge in this question is defining the right set of rules.

Conjecture (Lipton). Under a certain set of axioms, one cannot define progress towards computing f on a circuit.

We might arrive at a possible counterexample as follows: choose a random function $r : \{0, 1\}^n \rightarrow \{0, 1\}$, construct circuits for r and $f \oplus r$, and combine the outputs of each circuit to obtain $f = r + (f \oplus r)$. Note that the function $f \oplus r$ is also a random function, so neither r nor $f \oplus r$ has anything to do with f . Therefore, only the last operation should be important.

4 Variance of Strategies

Consider a zero-sum game

1	-1
-1	1

where player 1 selects a row and player 2 selects a column. If a player always picks the same row or column, the other player can always make a decision to win the game, so the best strategy is for each player to randomly choose.

Von Neumann showed that it is always possible for the two players to devise an optimal strategy. Consider now the game

0.9	-1
-1	0.9
10^6	-10^6
-10^6	10^6

where once again player 1 selects a row, and player 2 selects a column. Von Neumann's strategy would suggest that player one randomly play either rows 3 or 4. But in reality many people would not accept this variability, and would rather play a game that is slightly biased against them. Even if this was a zero-sum game, many players would opt only to play the first two rows.

The existing theory fails to account for this large variability.

Open Question. How do we incorporate variance into the problem? In general, can we control the higher moments? What is the complexity of such a model?