The Knights' Problem: DNA Algorithms for NP-complete Problems

Recall that SAT as we usually see it is the problem of determining whether or not there exists a way to assign variables so that a Boolean formula is true. We call this a **decision problem**.

In the Knights' Problem paper, the Boolean formula is this:

((¬h & ¬f) or ¬a) & ((¬g & ¬i) or ¬b) & ((¬d & ¬h) or ¬c) & ((¬c & ¬i) or ¬d) & ((¬a & ¬g) or ¬f)

1. Solve the decision problem: Is this formula satisfiable?

   Yes, it is satisfiable! For example, if there are no knights on the board (all variables are "False"), there is no possible arrangement in which they are attacking.

The Knights' Problem paper describes a DNA-based algorithm which does more than this: it *figures out* the variable assignments instead of simply saying True/False about whether such an assignment exists. The DNA-based algorithm solves the **search problem**: it *searches* for possibilities, and reports them.

> *Technical note about the definition of "search problems": An algorithm only needs to output **one** solution for us to say "the algorithm solves the search problem." It is not guaranteed to output all of them. The DNA-based algorithm tries for all of them; but when I say "the answer to the search version of SAT", assume that you only get to see **one** answer, not all of them.*

2. True, false, or open: You could use the answer to the *search* version of SAT to find the answer to the *decision* version of SAT, in polynomial time.

   True. If the search gives you something – then True! If not, then False!

3. True, false, or open: You could use the answer to the *decision* version of SAT to find the answer to the *search* version of SAT, in polynomial time.

   Open. (True if P=NP, False otherwise.) If you know the decision problem, all you know is True or False – not very helpful. The search version is still NP-complete, so a polynomial time algorithm for it only exists if P=NP.

4. Which would you expect to be more difficult: the *search* version or the *decision* version of SAT?
   As per the previous two questions, search is more difficult. With search you can solve decision; but with decision you probably can't solve search.

Suppose, instead, that I told you to find a valid Knights' Problem positioning of knights on the chessboard that uses the *most* knights. This is an **optimization problem**: you need to search for the "best" possibility (for some definition of "best"), and report it.

5. True, false, or open: You could use the answer to the *optimization* version of SAT to find the answer to the *decision* version of SAT, in polynomial time.
   True.

6. True, false, or open: You could use the answer to the *optimization* version of SAT to find the answer to the *search* version of SAT, in polynomial time.
   True.

7. Which of the following are currently known to be in P: search version of SAT, decision version of SAT, optimization version of SAT?
   None of these.

8. Which of the following are definitely in NP (think about what the polynomial-time 'proof' could be): search version of SAT, decision version of SAT, optimization version of SAT?
   Decision: definitely in NP (proof: the satisfying solution)
   Search: definitely in NP (proof: the satisfying solution)
   Optimization: not sure. I don't think it is. You'd have to show all 2^n possibilities to prove that you've found the best one.

9. Which of the following are NP-hard: search version of SAT, decision version of SAT, optimization version of SAT?

10. If you were to arrange these three types of problems – *decision, optimization, and search* – in order from hardest to easiest, how would you arrange them?
    Optimization is harder than search. Search is harder than decision.

Is DNA computing faster?

For a bit, let us assume that we have infinite lab space and DNA nucleotides to use. We'll focus on the *time* that algorithms take to run, focusing on the *search version* of the Knights' Problem.

If we were working on a classical computer, we could use the "brute force" algorithm to solve a Boolean formula: try *all* possible variable assignments until one works.

1. If the Boolean formula is length x, how much time does the computer algorithm take to find a satisfying solution: logarithmic, linear, polynomial, or exponential in x?
   exponential

2. If the Boolean formula is length x, then how many lab steps does the DNA algorithm need to find a satisfying solution: logarithmic, linear, polynomial, or exponential in x?

   (Assume that diffusion and reactions are infinitely fast, and that the only slow steps of the algorithm are the lab work: mixing things together, adding the enzyme, etc.)
   polynomial

3. Are there any problems with my assumption that diffusion and reactions are infinitely fast, when we're working with massive quantities of DNA?
   yes (elaborate)

Scalability of DNA Computing

Let's think about expanding the 3x3 Knights' Problem up to an 8x8 board! First, let's try to calculate the *space* needs of the DNA algorithm.

1. How many *variables* (a, b, c, …) do you need to represent this problem?
   64, of course.

2. How many different DNA sequences will be in the DNA library that you need to make?
   2^{64}

3. Assume that you need at least 30 nucleotides to represent each variable. How long (in number of nucleotides) does each DNA strand need to be, in total?
   30*64

4. Combining your answers to (2) and (3), how many *moles* of DNA *nucleotides* do you need as ingredients to synthesize your DNA? There are 6.02×10^{23} molecules in a mole.
   30*64 nucleotides per sequence * 2^{64} sequences * 1/ 6.02×10^{23} moles per nucleotide
   = 0.05883346947 mol

5. On average, a DNA nucleotide weighs 330 grams per mole. How many grams of DNA do you need?
   0.05883346947 mol * 330 g/mol = 19.4150449254 g

6. How does the amount of DNA that you need grow as the area of the chessboard grows: logarithmically, linearly, polynomially, or exponentially?
   (exponentially)

7. Consider, instead, a 9x9 chessboard. Would the experiment be feasible?
   Hahahaha no. The number that you get out at the end becomes 3 million grams. Note that each person has around 10 grams of DNA in their entire body. 3 million grams is also equivalent to the average body weight of 40 people.