B Filtering models

Filtering models, an important class of DNA algorithms, operate by filtering out of the solution molecules that are not part of the desired result. A chemical solution can be treated mathematically as a finite *bag* or *multi-set* of molecules, and filtering operations can be treated as operations to produce multi-sets from multi-sets. Typically, for a problem of size n, strings of size $\mathcal{O}(n)$ are required. The chemical solution should contain enough strings to include many copies all possible answers. Therefore, for an exponential problem, we will have $\mathcal{O}(k^n)$ strings. Filtering is essentially a brute-force method of solving problems.

B.1 Adleman: HPP

B.1.a Review of HPP

Leonard Adleman's solution of the Hamiltonian Path Problem was the first successful demonstration of DNA computing. The *Hamiltonian Path Problem* (HPP) is to determine, for a given directed graph $G = (V, E)$ and two of its vertices $v_{\text{in}}, v_{\text{out}} \in V$, whether there is a *Hamiltonian path* from v_{in} to v_{out}, that is, a path that goes through each vertex exactly once. HPP is an NP-complete problem, but we will see that for Adleman's algorithm the *number of algorithm steps* is linear in problem size.

Adleman (the "A," by the way, of "RSA.") gave a laboratory demonstration of the procedure in 1994 for $n = 7$, which is a very small instance of HPP. (We will use this instance, shown in Fig. IV.7, as an example.) Later his group applied similar techniques to solving a 20-variable 3-SAT problem, which has more than a million potential solutions (see p. 215).[4]

B.1.b Problem Representation

The heart of Adleman's algorithm is a clever way to encode candidate paths in DNA. Vertices are represented by single-stranded 20mers, that is, sequences of 20nt (nucleotides). They were generated at random and assigned to the vertices, but with the restriction that none of them were too similar or complementary. Each vertex code can be considered a catenation of two 10mers: $v_i = a_i b_i$ (i.e., a_i is the 5′ 10mer and b_i is the 3′ 10mer). Edges are also

[4] `https://en.wikipedia.org/wiki/Adleman,_Leonard` (accessed 2012-11-04).

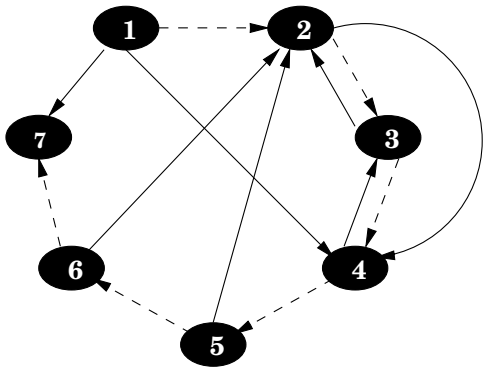

Figure IV.7: HPP solved by Adleman. The HP is indicated by the dotted edges. [source: Amos, Fig. 5.2]

---

represented by 20mers. The edge from vertex i to vertex j is represented by

$$e_{i\to j} = b_i a_j, \text{ where } v_i = a_i b_i, \text{ and } v_j = a_j b_j.$$

Paths are represented by using complements of the vertex 20mers to stitch together the edge 20mers. (Of course, using the complements of the edges to stitch together the vertices works as well.) For example, a path $2 \to 3 \to 4$ is represented:

$$\begin{array}{cccc} \overbrace{b_2 \quad a_3}^{e_{2\to 3}} & & \overbrace{b_3 \quad a_4}^{e_{3\to 4}} & \\ \hline & \underbrace{\overline{a_3} \quad \overline{b_3}}_{\overline{v_3}} & & \end{array}$$

The edges from v_{in} and to v_{out} have special representations as 30mers:

$$\begin{aligned} e_{\text{in}\to j} &= v_{\text{in}} a_j, \text{ where } v_j = a_j b_j, \\ e_{i\to\text{out}} &= b_i v_{\text{out}}, \text{ where } v_i = a_i b_i. \end{aligned}$$

Note that the special representation of the initial and terminal edges results in blunt ends for complete paths.

Therefore, for the $n = 7$ problems, candidate solutions were 140bp in length: There are $n-1$ edges, but the first and last edges are 30mers. Hence $2 \times 30 + (n-3) \times 20 = 140$. Ligation is used to remove the nicks in the backbone.

B.1.c Adleman's Algorithm

algorithm Adlemen:

Step 1 (generation of all paths): Generate multiple representations of all possible paths through the graph. This is done by combining the oligos for the edges with the oligos for the complements of the vertices in a single ligation reaction.

Step 2: Amplify the concentration of paths beginning with v_{in} and ending with v_{out}. This is done by PCR using v_{in} and $\overline{v_{\text{out}}}$ as primers. Remember that denaturation separates the sense and antisense strands. PCR extends the sense strand in the $3'$ direction from v_{in}, and extends the antisense strand in the $3'$ direction from $\overline{v_{\text{out}}}$. At the end of this step we have paths of all sorts from v_{in} to v_{out}.

Step 3: Only paths with the correct length are retained; for $n = 7$ this is 140bp. This operation is accomplished by gel electrophoresis. The band corresponding to 140bp is determined by comparison with a *marker lane*, and the DNA is extracted from this band and amplified by PCR. Gel electrophoresis and PCR are repeated to get a sufficient quantity of the DNA. We now have paths from v_{in} to v_{out}, but they might not be Hamiltonian.

Step 4 (affinity purification): Select for paths that contain all the vertices (and are thus necessarily Hamiltonian). This is done by first selecting all those paths that contain v_1, and then, of those, all that contain v_2, and so forth. To select for v_i, first heat the solution to separate the strands. Then add the $\overline{v_i}$ bound to a magnetic bead. Rehybridize (so the beads are bound to strands containing v_i), and use a magnet to extract all the paths containing v_i. Repeat this process for each vertex.

Step 5 If there any paths left, they are Hamiltonian. Therefore amplify them by PCR and inspect the result by gel electrophoresis to see if there are any

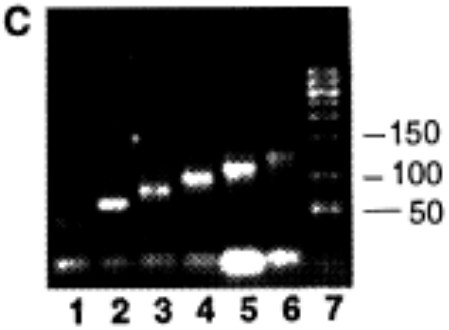

Figure IV.8: Electrophoresis showing solution to HPP problem.

strands of the correct length. If there are, then there is a Hamiltonian path; if there aren't, then there is not. If desired, the precise HP can be determined by a graduated PCR procedure: Run $n-1$ parallel PCR reactions. In the ith lane, v_{in} is the left primer and v_i is the right primer. This will produce bands with lengths 40, 60, 80, 100, 120, and 140 bp. The lane that has a band at 40 corresponds to the first vertex after v_{in} in the path, the lane with a band at 60 corresponds to the next vertex, etc. This final readout process depends on there being only one Hamiltonian path, and it is error-prone due to its dependence on PCR.
□

B.1.d Discussion

Adleman's algorithm is linear in the number of nodes, since the only iteration is Step 4, which is repeated for each vertex. Step 5 is also linear if the path is read out. Thanks to the massive parallelism of molecular computation, it solves this NP-complete problem in linear time. Adleman's experiment took about a week, but with a more automated approach it could be done in a few hours. On the other hand, the PCR process cannot be significantly shortened.

In addition to time, we need to consider the *molecular resources* required. The number of different oligos required is proportional to n, but the number of strands is much larger, since there must be multiples instances of each

possible path. If d is the average degree of the graph, then there are about d^n possible paths (exponential in n). For example, if $d = 10$ and $n = 80$, then the required 10^{80} DNA molecules is more than the estimated number of atoms in the universe. Hartmanis calculated that for $n = 200$ the weight of the DNA would exceed the weight of the earth. So this brute-force approach is still defeated by exponential explosion. Lipton (1995) estimates that Adleman's algorithm is feasible for $n \leq 70$, based on an upper limit of $10^{21} \approx 2^{70}$ DNA strands (Boneh, Dunworth, Lipton & Sgall, 1996), but this is also feasible on conventional computers.

Nevertheless, Adleman's algorithm illustrates the massive parallelism of molecular computation. Step 1 (generation of all possible paths) took about an hour for $n = 7$. Adleman estimates that about 10^{14} ligation operations were performed, and that it could be scaled up to 10^{20} operations. Therefore, speeds of about 10^{15} to 10^{16} ops/sec (1–10 peta-operations/s) should be achievable, which is, digital supercomputer range. Adlemen also estimates that 2×10^{19} ligation operations were performed per joule of energy. Contemporary supercomputer perform only 10^9 operations per joule, so molecular computation is 10^{10} more energy-efficient. It is near the thermodynamic limit of 34×10^{19} operations per joule. Recall (Ch. II, Sec. B.1) $kT \ln 2 \approx 3 \times 10^{-9}$pJ $= 3 \times 10^{-21}$J, so there can be about 3.3×10^{20} bit changes/J.[5]

A more pervasive problem is the inherent error in the filtering processes (due to incorrect hybridization). Some strands we don't want, get through; and some that we do want, don't. With many filtering stages the errors accumulate to the extent that the algorithms fail. There are some approaches to error-resistant DNA computing, but this is an open problem.

[5]DNA is of course space efficient. One bit of information occupies about 1 cubic nm, whereas contemporary disks store a bit in about 10^{10} cubic nm. That is, DNA is a 10^{10} improvement in density.