B Filtering models

¶1. **Filtering models:** Operate by filtering out of the solution molecules that are not part of the solution.

¶2. A 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.

¶3. **Initial multi-set:** Typically, for a problem of size n, strings of size $\mathcal{O}(n)$ are required.
Should contain enough strings to include many copies all possible solutions.
Therefore, for an exponential problem, we will have $\mathcal{O}(k^n)$ strings.

¶4. This is essentially a brute-force method.

B.1 Adleman: HPP

B.1.a Review of HPP

¶1. **Hamiltonian Path Problem (HPP):** The *Hamiltonian Path Problem* 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 HP from v_{in} to v_{out}, that is, a path that goes through each vertex exactly once.

¶2. **NP-complete:** HPP is an NP-complete problem.

¶3. We will see that for Adleman's algorithm the *number of algorithm steps* is linear in problem size.

¶4. **Laboratory demonstration:** Leonard Adleman gave a laboratory demonstration of the procedure in 1994 (for $n = 7$).
(By the way, he is the "A" of "RSA.")

¶5. "In 2002, he and his research group managed to solve a 'nontrivial' problem using DNA computation. Specifically, they solved a 20-variable SAT problem having more than 1 million potential solutions.

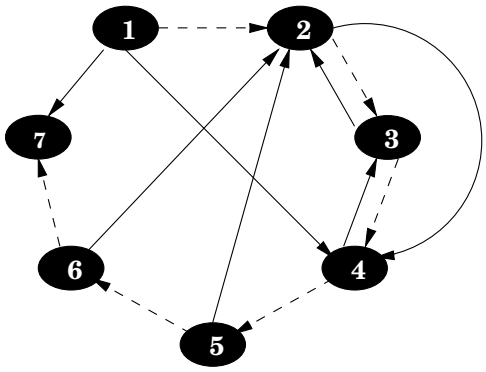


Figure IV.6: HPP solved by Adleman. The HP is indicated by the dotted edges. [source: Amos, Fig. 5.2]

They did it in a manner similar to the one Adleman used in his seminal 1994 paper."[4]

B.1.b Problem Representation

¶1. The heart of Adleman's algorithm is a clever way to encode candidate paths in DNA.

¶2. **Vertices:** Vertices were represented by single-stranded 20mers, that is, sequences of 20nt (nucleotides).

¶3. They were generated at random and assigned to the vertices, but with the restriction that none of them were too similar or complementary.

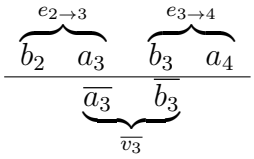
¶4. Each vertex code can be considered a catenation of two 10mers: $v_i = a_i b_i$.
(I.e., a_i is the $5'$ 10mer and b_i is the $3'$ 10mer.)

¶5. **Edges:** Also represented by 20mers.
The edge from vertex i to vertex j is represented by

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

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

¶6. **Paths:** 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:



¶7. **Initial and final edges:** Edges from v_{in} and to v_{out} have special representations as 30mers:

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

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

¶9. 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 $2 \times 30 + (n - 3) \times 20 = 140$.

¶10. Ligation is used to remove the nicks in the backbone.

B.1.c Algorithm

¶1. **Step 1:** Generate multiple representations of all possible paths through the graph.

¶2. This is done by combining the oligos for the edges with the oligos for the complements of the vertices in a single ligation reaction.

¶3. **Step 2:** Amplify the concentration of paths beginning with v_{in} and ending with v_{out}.

¶4. 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 moves in the 3′ direction from $\overline{v_{\text{in}}}$ on the sense strand,
and in the 3′ direction from v_{out} on the antisense strand.

¶5. We now have paths of all sorts from v_{in} to v_{out}.

¶6. **Step 3:** Only paths with the correct length are retained. For $n = 7$ this is 140bp.

¶7. This is accomplished by gel electrophoresis.
The band corresponding to 140bp is determined by comparison with a *marker lane*.
The DNA is extracted from this band and amplified by PCR.

¶8. Gel electrophoresis and PCR are repeated to get a sufficient quantity.

¶9. We now have paths from v_{in} to v_{out}, but they might not be Hamiltonian.

¶10. **Step 4 (affinity purification):** Select for paths that contain all the vertices (and are thus necessarily Hamiltonian).
This is done by selecting all those that contain v_1,
and of those, all that contain v_2, and so forth.

¶11. To do select for v_i, first heat the solution to separate the strands.
Then add the $\overline{v_i}$ bound to a magnetic bead.
Rehybridize, and use a magnet to extract all the paths containing v_i.
Repeat for each vertex.

¶12. **Step 5:** If there any paths left, they are Hamiltonian.
Therefore amplify them by PCR.
Inspect the result by gel electrophoresis to see if there are any strands of the correct length.
If there are, then there is a HP; if not, then not.

¶13. **Determining the path:** The precise HP can be determined by a graduated PCR procedure.
Run $n - 1$ PCR reactions.
In the ith lane, v_{in} is the left primer and v_i is the right primer.
"[T]he unique HP ... should produce bands of length 40, 60, 80, 100, 120, and 140 b.p. in lanes 1 through 6, respectively." [Amos, p. 114]
That is, the primer that produced 40 is the second vertex in the HP; the primer that produced 60 is the third, etc.

¶14. The final process depends on there being only one HP and is error-prone due to its dependence on PCR.

B.1.d Discussion

¶1. **Linear:** Adleman's algorithm is linear in the number of nodes, since the only iteration is Step 4, which is repeated for each vertex.

¶2. Adleman's experiment took about a week, but with a more automated approach it could be done in a few hours.

¶3. On the other hand, the PCR process cannot be significantly shortened.

¶4. **Molecular resources:** The number of different oligos required is proportional to n.

¶5. **Strands:** The number of strands is much larger, since there must be a number of representatives 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 10^{80} DNA molecules is more than the estimated number of atoms in the universe.

¶6. Hartmanis calculated that for $n = 200$ the weight of the DNA would exceed the weight of the earth.

¶7. So this brute-force approach is still defeated by exponential explosion.

¶8. Lipton (1995) estimates that it is feasible for $n \leq 70$, but this is also within the range of conventional computer.

¶9. **Massive parallelism:** 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.
That is, digital supercomputer range.

¶10. **Energy:** Adlemen estimates that 2×10^{19} ligation operations were performed per joule of energy.
Contemporary supercomputer perform only 10^9 operations per joule, so MC is 10^{10} more energy efficient.

¶11. It is near the thermodynamic limit of 34×10^{19} operations per joule.
Recall (Ch. II, Sec. B.1) $kT \ln 2 \approx 3 \times 10^{-9}\text{pJ} = 3 \times 10^{-21}\text{J}$, so there can be about 3.3×10^{20} bit changes/J.

¶12. **Space:** One bit of information occupies about 1 cubic nm.
Contemporary disks store a bit in about 10^{10} cubic nm.
That is, a 10^{10} improvement in density.

¶13. **Inherent error:** 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.

¶14. There are some approaches to error-resistant DNA computing.