Modeling Non-specific Binding in Gel-Based DNA Computers

Clifford R. Johnson
clifford.johnson@usc.edu

Abstract. In attempting to automate the computation of n-variable 3-CNF SAT problems using DNA, two physical architectures were scrutinized, the "in-line" architecture and the "waste-well" architecture. Computer modeling of the effects of non-specific binding predicted that the in-line version would not work for problems of more than 7 variables. According to the model, the "wrong answer" DNA strands would swamp out the "correct answer" DNA strands in the final computation module. And in fact, the in-line architecture never performed a computation higher than 6 variables.

To perform a 20 variable instance of the 3-CNF SAT problem a manual version of the waste-well architecture was employed. Surprisingly though, after analysis of the modeling results, it appears that through a simple protocol change, the in-line architecture may have been able to perform higher order computations.

1 Introduction

The first molecular computation was performed by Len Adleman at the University of Southern California in 1994. Using DNA molecules to perform the computation, Adleman solved a 7-city, Directed Hamiltonian Path Problem [1]. The molecular implementation used to solve the 7-city DHPP was surprisingly simple. However because of the use of enzymes (ligase) to covalently ligate strands, it was apparent that it would be almost impossible to scale-up the computation, i.e., to solve larger problems, using this paradigm. The making and breaking of covalent bonds using enzymes is notoriously inefficient; 40% reaction completion is considered good. It's messy, difficult, inefficient, error prone. Additionally, the envisioned simplicity of molecular computation disappears - computation schemes begin to look like Rube Goldberg devices. For computations on the order of a 20 variable problem (about 8,000 times more complex than the 7 city problem), a new molecular paradigm was necessary, one that could somehow avoid biology's inherent messiness.

A new paradigm was formulated based on Richard Lipton's method for encoding DNA to represent binary strings [2,3], and was called the "modified sticker model" [4]. It involves no enzymes, and no covalent bond formation or destruction. The computation is performed simply through the hybridization and denaturing of DNA hydrogen bonds. This is the paradigm used to perform the 20 variable 3-CNF SAT problem published in Science [5], which remains at this time, the most complex problem solved using molecules.

M.H. Garzon and H. Yan (Eds.): DNA 13, LNCS 4848, pp. 170–181, 2007.

For the most part, molecular computations are performed by hand, at the lab bench. The computation of the instance of the 20 variable 3-CNF SAT problem [5], took 2 to 3 man-weeks to perform by hand. This is labor intensive and error prone. One of the project goals was the automation of the computation process. In trying to automate the computation of 3-CNF SAT problems, the question arose: What *physical* hardware configuration is best? Two different architectures vied for the honor: One was called the "in-line" architecture; the other was called the "waste-well" architecture. Both architectures were actually implemented and tested.

Computer modeling of the effects of non-specific binding (NSB) predicted that the in-line version would not work for problems of more than 7 variables - the wrong "answer" DNA strands would swamp out the "correct answer" DNA strands in the final computation module. In fact, the in-line architecture never performed a computation higher than 6 variables, and the waste well architecture was employed to perform the 20 variable computation. Surprisingly though, after analysis of the modeling results, it appears that through a simple protocol change, the in-line architecture may have been able to perform higher order computations.

2 The 3-CNF SAT Problem

The Satisfiability problem (SAT) is of interest both historically and theoretically. Historically, the SAT problem was the first to be shown to be NP complete. Theoretically, the SAT problem plays a critical role in computer science applications and theory. In practice, the SAT problem is fundamental in solving many application problems in database design, CAD-CAM, robotics, scheduling, integrated circuit design, computer networking, and so on. "Methods to solve the SAT problem play a crucial role in the development of efficient computing systems." [6] SAT problems are a set of computationally intractable NP-complete problems. Problems in Class NP are considered intractable because as the number of variables increases linearly, the computation time increases exponentially. For example, a 100 variable instance of a 3-SAT problem might take IBM's Big Blue, computing at 135 teraFLOPS, 3.2 million centuries to solve, essentially the problem is unsolvable.

The SAT Problem

The goal of the SAT problem is to determine whether there exists a satisfying truth assignment for a given Boolean expression. That is:

Let $U = \{x_1, x_2, ..., x_n\}$ be a set of n Boolean variables. A truth assignment for U is a function $T : U \rightarrow$ {true, false}. Corresponding to each variable x_i are two literals, x_i and $\neg x_i$ (not x_i) that can be assigned to the variable. A literal x_i is `true` under T iff $T(x_i)$ = true; a literal $\neg x_i$ is `true` under T iff $T(x_i)$ = false**).**

A set of literals surrounded by parentheses is called a clause, and a set of clauses is called a formula.

A satisfying assignment for a formula, φ is called a solution.

The restriction of SAT to instances where all clauses have length k is called k-SAT.

The Conjunctive Normal Form (CNF)
Let φ be a formula. Let *C* be the set of clauses for that formula. φ is a formula in conjunctive normal form (CNF), implies that a truth assignment $T : U \rightarrow \{\text{true, false}\}$ satisfies $c \in C$ iff at least one literal in *c* is true under *T*. *T* satisfies φ iff it satisfies every clause in φ.

Equation 1 shows an instance of a 10 variable 3-CNF SAT problem with 14 clauses. Notice that there are 3 literals per clause separated by the OR symbol v, and that each clause is separated by the AND symbol ^. This is the conjunctive normal form for a formula 3-SAT problem.

$$\begin{aligned}\phi = {} & (X_2 \vee X_4 \vee X_9) \wedge (X_8 \vee \neg X_{10} \vee X_5) \wedge (\neg X_6 \vee \neg X_8 \vee \neg X_{10}) \wedge (X_2 \vee \neg X_4 \vee \neg X_9) \wedge \\ & (\neg X_9 \vee \neg X_3 \vee X_6) \wedge (X_{10} \vee X_5 \vee X_7) \wedge (\neg X_7 \vee X_1 \vee \neg X_2) \wedge (X_2 \vee \neg X_4 \vee X_9) \wedge \\ & (X_3 \vee X_6 \vee \neg X_8) \wedge (\neg X_5 \vee X_7 \vee X_1) \wedge (\neg X_2 \vee \neg X_4 \vee \neg X_9) \wedge (X_2 \vee X_4 \vee \neg X_9) \wedge \\ & (\neg X_1 \vee \neg X_2 \vee X_4) \wedge (X_2 \vee \neg X_4 \vee X_9)\end{aligned} \quad (1)$$

Here ϕ has the unique solution:
$X_1 = F, X_2 = T, X_3 = T, X_4 = F, X_5 = F, X_6 = F, X_7 = F, X_8 = T, X_9 = F, X_{10} = T.$

To solidify these concepts in an informal fashion, think of this as a kind of Agatha Christie murder mystery. Ten professors, named Professor X1, Professor X2, ..., and Professor X10, are invited to dinner. Some of the professors may have been "eliminated" on their way to dinner. We want to know who made it to the dinner, and who didn't. The clauses provide clues. For example, the first clause tells us that either *Professor X2 arrived,* ***OR*** *Professor X4 arrived,* ***OR*** *Professor X9 arrived* for dinner that night. The second clause, for example, tells us that either *Professor X8 arrived,* ***OR*** *Professor* X_{10} *did not arrive,* ***OR*** *Professor X5 did arrive* for dinner.

If we put all of the clues (clauses) together, we get the solution to the mystery. In the unique solution for ϕ for Equation 1, we see that Professor X1 did *not* arrive to dine, whereas Professor X2 did, and so on. Here is the interesting part. If one were to try to solve Equation 1, without knowing the answer beforehand, it would take a very long time to find the solution, even for this relatively short 10 variable problem. Yet once we are given a solution for ϕ, it is very easy to verify. We just check to see if at least one literal in each clause is true. (This can be seen with Equation 1.) This is the essence of Class NP problems. Problems in Class NP are very, very hard to solve. Yet once a solution is found, it can be *verified* quickly.

3 The Molecular Implementation of the 3-CNF SAT Problem

The implementation paradigm is remarkably straightforward:

1. To represent all possible variable assignments for the chosen n-variable SAT problem, a Lipton encoding [7] for DNA strands is chosen. For each of the n variables $x_1, x_2, \ldots, x_n$, two distinct 15 base value sequences are designed - one representing *true (T),* X_kT, and one

representing *false (F), X_kF.* Each of the 2^n truth assignments is represented by a sequence of (*n* X 15) bases consisting of the concatenation of one value sequence for each variable. In this way all possible assignments are encoded. DNA molecules with library sequences are termed library strands; a combinatorial pool containing library strands is termed a library.

2. The probes used for separating the library strands have sequences that are W-C (Watson-Crick) complements of the value sequences.
3. The *clauses* of an n-var CNF SAT problem are formed with acrylamide gel modules in which the probes for the clause are covalently bonded to the acrylamide gel. For example, for the last clause of Eq. 1, (X_2 v $\neg X_4$ v X_9) the W-C complementary probes for X_2T , X_4F, X_9T are covalently bound to the acrylamide gel. (Figure 1.)

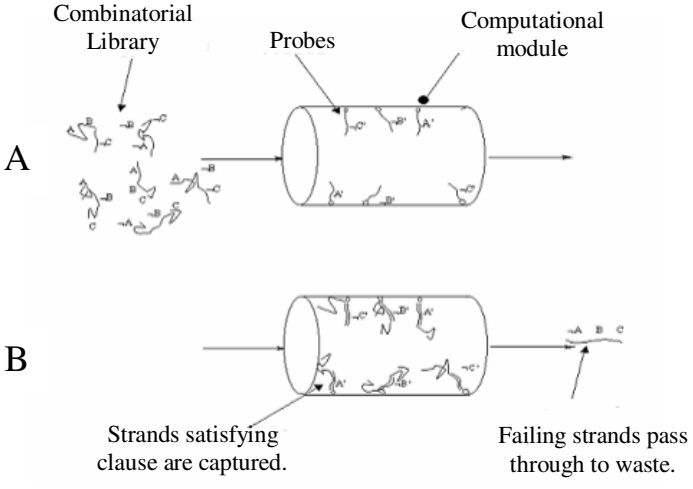

Fig. 1. A and B. A computation. A The combinatorial library enters the gel module which tests the clause (X_2 v $\neg X_4$ v X_9) using covalently bound DNA probes that are W-C complimentary to X_2, $\neg X_4$, and X_9. B Strands that do not satisfy the clause, pass through to waste.

4 The Physical Architectures

The Molecular Algorithm

1. Under hybridizing conditions (temperature at 15° C), introduce the DNA strand library into the first module via electrophoresis. (This library represents all possible variable assignments.)
2. Under hybridizing conditions, those strands that satisfy the clause, hybridize to the probes and remain in the gel module. Those strands that do not satisfy the clause pass through.

3. The gel module is then heated to 65 ° C to release the hybridized (satisfying) strands, which are then passed via electrophoresis to the next cooled module and a new computation.
4. The strands captured in the final module represent those variable assignments that have successfully satisfied all of the clauses and thus represent a solution.
5. The final gel module is removed to extract the DNA molecules for PCR amplification and sequencing of the answer.

Note that this molecular algorithm for the SAT computation is massively parallel.

Two different architectures were considered as candidates for automating this algorithm for solving CNF SAT computations using DNA - the in-line architecture and the waste-well architecture.

The In-Line Architecture

The premise of this geometry (Fig. 2) is that as long as the next module down stream is heated, a valid computation can be performed. Fig. 3 diagrams the in-line device set-up during a computation. In Fig. 3, capture Module 1 is heated to release the combinatorial library, which then moves via electrophoresis to Module 2. The cooled Capture Module 2 captures those strands satisfying its clause while those strands not satisfying the clause pass through to the next consecutive module (Module 3), which is held hot. Theoretically, no strands should remain in the adjacent down-stream module (Module 3) as the temperature is raised to the same release temperature as in Module 1. All non-satisfying strands continue on to Module 4 (which is at room temperature) theoretically clearing out Module 3. As the Hot-Cold-Hot manifold moves to the right performing computations, all of the non-satisfying strands will eventually empty to the waste reservoir, in theory.

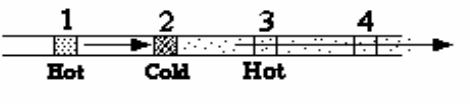

Fig. 2. Schematic of the in-line architecture

Note that all of the library strands pass through all of the computation modules in this configuration. It is assumed that the heating of down-stream modules is sufficient to release all of the oligonucleotides residing therein and will cause no contamination in the upcoming computation. In short, NSB (non-specific binding) is assumed to be inconsequential.

Though a 6-variable problem was solved using the above set-up, the in-line architecture didn't work for larger *n*'s. Trying to solve a 10-variable problem using the in-line architecture, proved fruitless; and in fact, to solve the 20-variable problem later on, an un-automated version of the waste-well architecture was employed - it was necessary to perform each of the computations by hand.

However, contradicting the assumption that NSB was inconsequential was some experimental evidence that detectable levels of NSB were indeed present in the gels. Computational test runs were performed using ^{32}P labeled library. The library was sent through a series of 24 sample modules, using an in-line construction similar to the one in Fig. 2. Some of these modules acted as capture/release modules; some

were simple agarose modules; and some simple acrylamide modules. At the end of the test, all had some level of residual radioactivity as measured with a Beckman Scintillation Counter. The residual radioactivity varied from 0.8% of the total counts to 6.5% of the total counts. When gels were removed from the glass modules (generally the gels slid out very easily), it was determined that about ¼ of the radiation was retained in the glass module, even after squirting distilled water through the gel trough to remove contaminants. The interpretation of these results was: (1) NSB was occurring on the glass surface, and (2) some sort of NSB / oligonucleotide retention was occurring in the gel itself. It is not known what the mechanism is for the retention of oligonucleotides in the gel. Inclusions, micro-fissures, poor gel formation, impurities, or some sort of bonding with the gel, any or all might be responsible for the phenomenon. Some of the radioactivity was probably due to radioactive mononucleotides. However, assuming that NSB is inconsequential is a problematic premise.

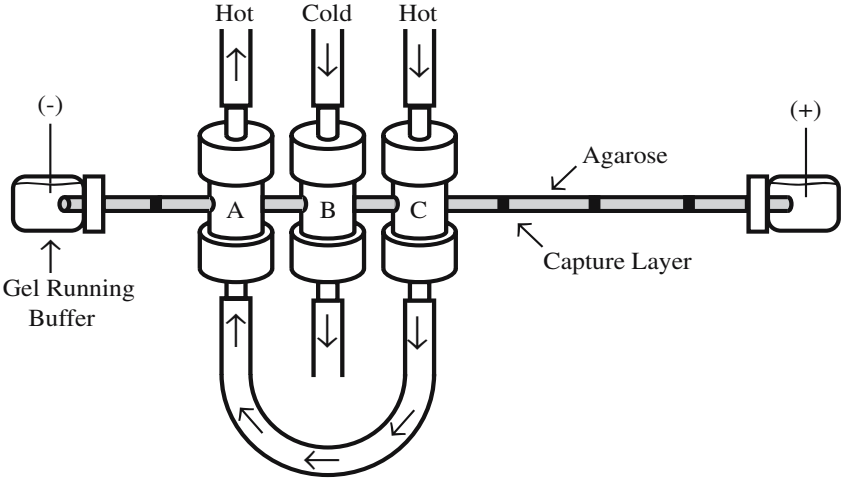

Fig. 3. The In-line Architecture. A 35-cm glass tube loaded with the library module, then with intercalated blank gel modules, and clause modules. The system was fitted with three water jackets (A, B, C). Library strands in the capture layer inside of (A) are released and move into the capture layer inside of (B). There, library strands with subsequences complementary to the probes are captured and retained. The rest of the strands passed into the capture layer inside of (C) but because (C) is kept hot the strands passed through unhindered.

The advantage of the in-line architecture is its simplicity. It is basically a glass tube packed with computation modules at equally spaced intervals, intercalated by gel. To automate the architecture one simply moves a Hot-Cold-Hot manifold down the glass tube (refer to Fig. 3). The disadvantage of the in-line system is that all library strands, "good" strands and "bad," go through every computation module, thus possibly contaminating the modules.

The Waste-Well Architecture

The second geometry (Fig. 4) is called the waste-well geometry. In this architecture, the non-satisfying strands of a computational step avoid passing through every downstream module by going to a waste buffer well, where they are destroyed. Again, strands are released from module 1, which then pass through module 2. Those strands that satisfy the Module 2 clause are captured and those that do not pass through the module. However, instead of continuing downstream possibly contaminating pristine modules, they are diverted immediately to a waste well, where they are destroyed. This second geometry was specifically conceived to obviate the accumulating effects of NSB. The waste well architecture is not as simple as the in-line architecture but it does preclude the effects of NSB.

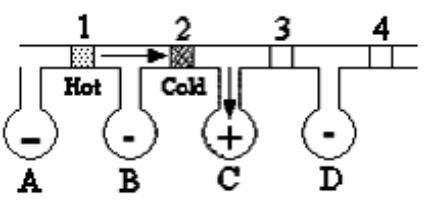

Fig. 4. Schematic of the "waste-well" architecture

This paradigm forms the architecture for the first functional automated molecular computer [8] solving instances of 10 variable 3-CNF SAT problems.

The advantage of the waste-well architecture is that unsuccessful DNA strands go to waste immediately after the computation, thus leaving downstream modules pristine. The disadvantage of the waste-well architecture is that it is complicated to construct [8].

5 The Mathematical Model

A priori, modeling the adsorption (i.e., NSB) of DNA in a gel based system would seem very difficult. First, one would have to determine the dominant forces involved in the binding reaction both to the gel and to the silica. For silica, some studies indicate that three effects, namely: (i) shielded intermolecular electrostatic forces, (ii) dehydration of the DNA and silica surfaces, and (iii) intermolecular hydrogen bond formation in the DNA–silica contact layer, are the dominant contributors to adsorption.[9] For gels, which are mostly fluid, a balance of forces maintains the gel form (sometimes even disturbing them infinitesimally can bring on a phase transitions and/or collapsing of the gel).[10] As mentioned above, inclusions, micro-fissures, poor gel formation, impurities, or some other sort of bonding of DNA with the gel, might contribute to NSB.

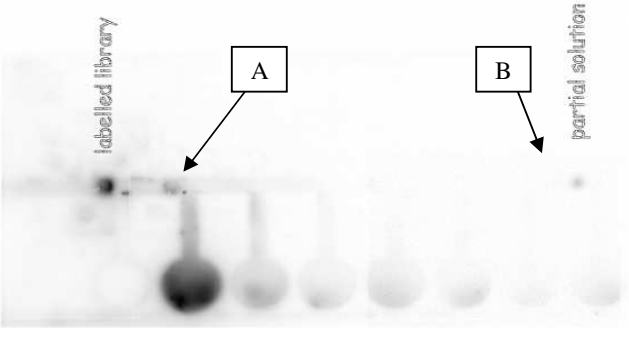

Fig. 5. Composite radioactive image showing prog-ression of a computation

These factors would make modeling of NSB a virtual nightmare. However, by using an *output/input* model, and describing the difference of *input - output* as due to NSB, one can arrive at a useful model that seems to be consistent with experimental data. To determine the ratio of molecules left behind in each computational module, that is:

$$(input\text{-}output)/input$$

experimental data is needed.

Fig. 5 was obtained using a Storm phosphor imaging sys-tem. The progression of a computation using lib-rary labeled with ^{32}P was imaged. Arrow A points to the first module in the computation. Here, the heavy residual radio-activity in this module is probably due to radio-active mononucleotides, and not to NSB of the library strands. Thereafter, residual radiation dropped drastically; but there was always a slight amount left. This "slight amount" was deemed to be due to NSB of library strands, and ranged in value from .05% to 1% (barely visible in the above image). Arrow B points to the 7th module with a NSB of about 0.1%. To the right of the module we can just barely see a partial solution progressing through a computation.

Modeling the In-Line Architecture

The model uses the following two assumptions:

1. Non-specific binding takes place in cold modules and has a very low constant of disassociation; i.e., it takes hours instead of seconds for NSB disassociation to occur, even at the elevated temperatures used to denature the probes from the library strands (65ºC).
2. Both complimentary and non-complimentary strands bind non-specifically with equal rates.

Both assumptions are reasonable. Assumption 1 ignores NSB in hot modules, yet it is apparent from experimental data that once NSB takes place, it takes hours for those strands to become disassociated.

The model is a set of linear difference equations that take into account binding and dissociation ratios under various conditions. The simulation was run on a spread sheet (Excel).

In the model, the integers *k, n, i* refer to the following:

k refers to the number of variables in the computation;
n refers to the computation step that is in progress;
i refers to the capture module that is in progress;

Let $\mathbf{X}$, $\mathbf{Y}$ be 2^k vectors, the components of which represent percentages of concentrations of each truth assignment strand of the combinatorial DNA.

$\mathbf{X^i}$ - this vector represents the percentage of released strands entering module *i* after having left module *(i – 1)*.

$\mathbf{Y^i}$ - this vector represents the percentage of strands binding in module *i*.

We can represent the state of various quantities of interest for each *i* th module at the *n* th computational step as a series of linear equations:

1. $\mathbf{Y}^{i}_{n} = \mathbf{C}^{i}\ \mathbf{X}^{i}_{n}$ i > n; $\mathbf{Y}^{i}_{n}$ gives the percentage of strands complimentary to the probes of the *i*th module that actually bind to those probes. $\mathbf{C}^{i}$ is a 2^{k} x 1 matrix of binding efficiencies.
2. $\mathbf{X}^{i+1}_{n} = [\,\mathbf{1} - \mathbf{C}^{i}\,]\ \mathbf{X}^{i}_{n}$; $\mathbf{X}^{i+1}_{n}$ gives the percentage of strands that did not bind to the *i* th module and which will continue on to the (i th + 1) module.
3. $\mathbf{X}^{i+1}_{n+1} = \mathbf{R}^{i}\ \mathbf{Y}^{i}_{n}$; $\mathbf{R}^{i}$ is a 2^{k} x 1 matrix of release efficiencies; $\mathbf{X}^{i+1}_{n+1}$ gives the percentage of strands released from a capture module after it is heated.
4. $\mathbf{E}^{i}_{n} = \mathbf{H}^{i}\ \mathbf{X}^{i}_{n}$; $\mathbf{E}^{i}$ represents the percentage of strands that bind non-specifically to the *i*th module at the *n* th step. $\mathbf{H}^{i}$ is a 2^{k} x 1 matrix of non-specific binding efficiencies.
5. $\mathbf{E}^{Total} = \sum_{n=1}^{n=k+4} \mathbf{E}^{i}_{n} - \mathbf{Y}^{i=k+4}_{n=k+4}$; $\mathbf{E}^{Total}$ gives the total percentage of strands that have bound non-specifically in the final module.

Using Excel for the simulation, we get the following surface graph (Fig. 6) that shows the effects of NSB vs. various binding efficiencies for a 6 variable computation. If we assume that we need at least 10 correct solution strands to every 1 error strand to un-ambiguously PCR amplify the read out, we see that NSB will prevent the correct readout of an answer. In general, we see that with even very small amounts of NSB, the ratio of good strands to contaminating (bad) strands drops drastically. Binding efficiency - the efficiency with which strands bind to their proper complementary probe - contributes to the problem, but not by very much.

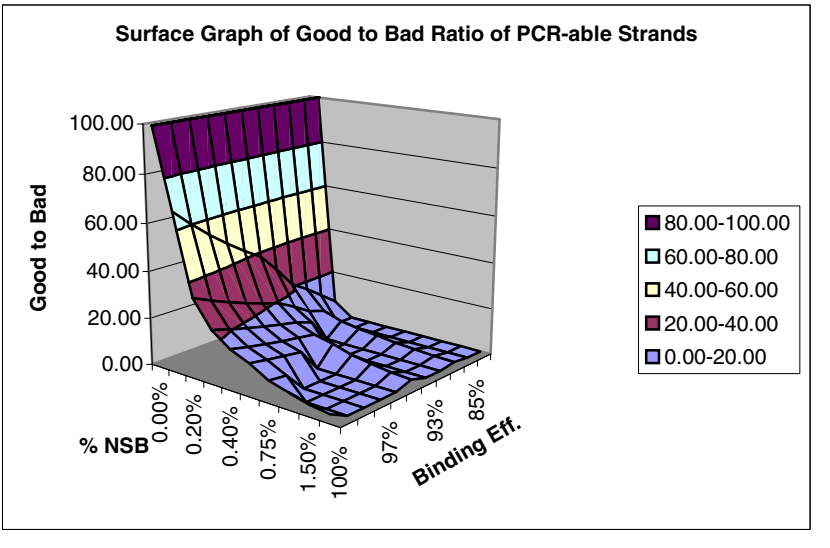

Fig. 6. Error Surface Graph

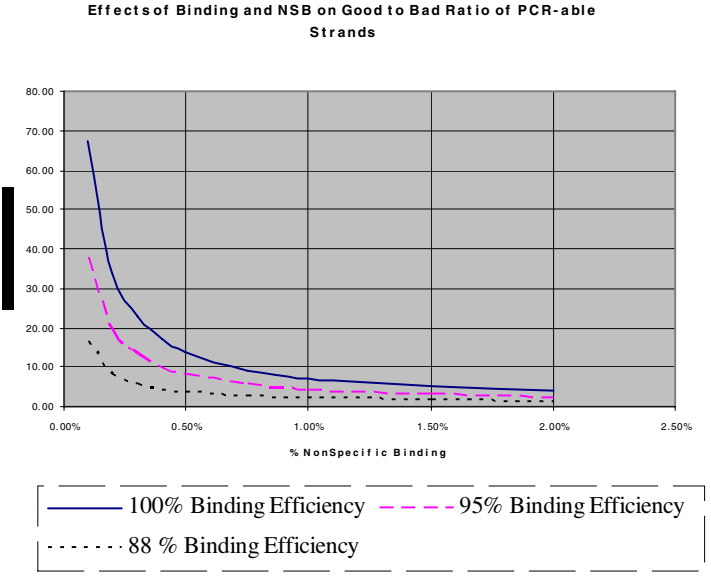

Fig. 7. Binding Efficiency Effect

This is seen more clearly in Fig. 7. In Fig. 7, we compare the ratios of correct answer strands to wrong strands found in the final module for three different binding efficiencies. 100% binding is the ideal, i.e., when all of the strands that should bind to probes in the final module do in fact bind; 95% binding is the efficiency claimed by the technical staff at Mosaic;[1] and 88% binding is the lowest efficiency obtained experimentally in the laboratory. As is seen in Fig. 7, for a 6 variable problem, any rate higher than 0.15% for NSB may cause problems in solution resolution. Fig. 8 extrapolates these results to problems of higher complexity.

Fig. 8 shows the results of the simulation for a NSB rate of 0.1%. The x-axis represents computational complexity, i. e., the number of variables in a 3-CNF SAT computation. From Fig. 8, we see that at the 0.1% rate, the percentage of NSB strands is equal to the percentage of answer strands for a 7 variable SAT problem. This corresponds closely with experimental observation.

Experimental results, phosphor imaging data and computation runs, are consistent with the modeling simulations - that is that the build up of contaminating strands due to NSB in the final module will swamp out the correct answer strands for computations with $n > 7$. So, if the model's premises are true, there seems to be no way to circumvent this build-up, and the in-line architecture seems to be condemned to toy computations of just a few variables.

Or is it?

The surprise lies in the way the final capture module is handled. From the DNA6 paper "Solution of a Satisfiability problem on a gel based DNA computer," [4]

[1] Personal communication.

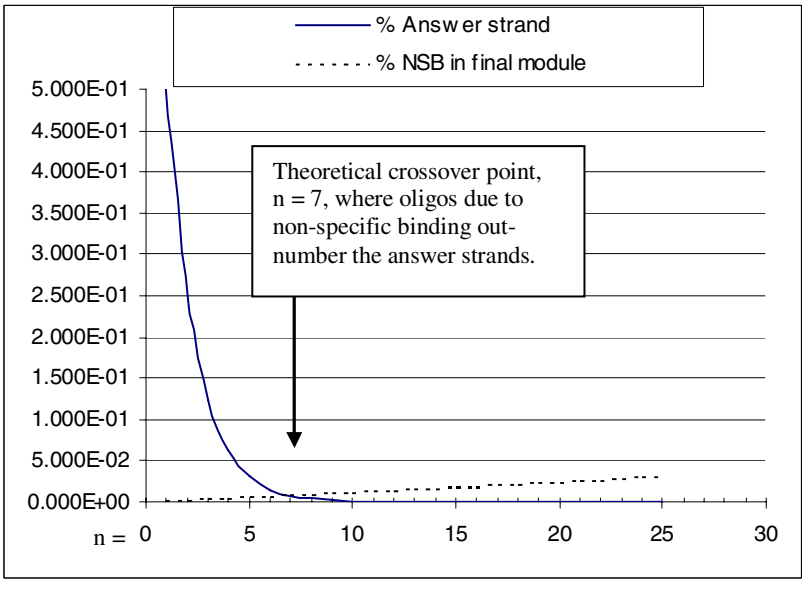

Fig. 8. Swamping Effect of Non-Specific Binding

we see that the gel is extracted from the glass tube and the final capture layer is dissected away. It is then crushed and soaked in 5 ml of water. The captured answer strands are then extracted from the gel by incubating the gel at 65ºC for 12 hours.

This procedure allows contaminating strands the time to leach out of the final gel module into solution along with the answer strands. Even though the dissociation time for NSB is on the order of hours, 65ºC for 12 hours is long enough for all DNA strands, answer strands as well as error strands, to be eluted from the crushed gel.

A better procedure would be, at the end of the computation, under denaturing conditions (65ºC), to elute the answer strands either into a pristine gel module or onto an elution membrane via electrophoresis, for half an hour. This would allow the answer strands cleared out of the final computation module leaving behind the error strands. Here, the long dissociation time for NSB works for us.

6 Conclusion

The in-line architecture is attractive because of its simplicity and the apparent ease of automatability. However, the in-line geometry was plagued by the effects of non-specific binding. We have seen that a simple protocol change would probably have lessened the effects of non-specific binding.

However, many different and poorly understood factors affect the phenomenon of non-specific binding – the type of gel, the buffer, the type of glass, can all affect non-specific binding. To properly characterize non-specific binding would be a lengthy and frustrating undertaking. In fact, the best strategy is probably to employ a geometry that precludes the effects of non-specific binding, that is, to use something

like the waste-well architecture that was eventually employed, albeit manually, in the 20 variable computation [5], and as an automated DNA computer in solving 10 variable SAT problems[8].

Acknowledgements

I would like to thank Rebecca A. Anderson for her support on this project.

References

1. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024 (1994)
2. Boneh, D., Dunworth, C., Lipton, R.: Breaking DES using a molecular computer. In: Lipton, R.J., Baum, E.B. (eds.) DNA Based Computers: Proceedings of a DIMACS Workshop, April 4, 1995. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 27, pp. 37–65. Princeton University. American Mathematical Society, Providence, RI (1996)
3. Lipton, R.: DNA Solution of Hard Computational Problems. Science 268(5210), 542–545 (1995)
4. Braich, R., Johnson, C., Rothemund, P.W.K., Hwang, D., Chelyapov, N., Adleman, L.: Satisfiability Problem on a Gel Based DNA Computer. In: Condon, A., Rozenberg, G. (eds.) DNA 2000. LNCS, vol. 2054, Springer, Heidelberg (2001)
5. Braich, R., Chelyapov, N., Johnson, C., Rothemund, P., Adleman, L.: Solution of a 20-Variable 3-SAT Problem on a DNA Computer. Science 296, 499–502 (2002)
6. Gu, J., Pardalos, P., Du, D. (eds.): Preface, Satisfiability Problem: Theory and Applications. DIMACS Series in Discrete Mathematics and Computer Science, American Mathematical Society, Providence, Rhode Island (1997)
7. Boneh, D., Dunworth, C., Lipton, R.: Breaking DES using a molecular computer. In: Lipton, R.J., Baum, E.B. (eds.) DNA Based Computers: Proceedings of a DIMACS Workshop, April 4, 1995. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 27, pp. 37–65. Princeton University. American Mathematical Society, Providence, RI (1996)
8. Johnson, C.: Automating the DNA Computer. In: Mao, C., Yokomori, T. (eds.) DNA Computing. LNCS, vol. 4287, Springer, Heidelberg (2006)
9. Melzak, K.A., Sherwood, C.S., Turner, R.F.B., Haynes, C.A.: Driving forces for DNA adsorption to silica in perchlorate solutions. J. of Colloid and Interface Science (181), 635–644 (1996)
10. Tanaka, T.: Gels. Scientific American 244(1), 124–138 (1981)