Making the SAT Decision Based on a DNA Computation

Joseph Ibershoff    Jerzy W. Jaromczyk    Danny van Noort

Abstract—Much of the recent research in DNA computing has focused on designing better overall techniques for computation, or implementing the techniques in simulation or a wet-lab in order to show the viability of these techniques for solving small SAT problems. In this paper, we examine a major obstacle to using DNA computing to solve larger, real-world SAT problems for which the correct answer is not already known. In particular, we ask the following question: Given the results of a DNA computation, how does one determine the answer to the underlying SAT problem, and how does one examine the confidence of this answer? We examine this question in detail for selection-based DNA computing, and show that it is non-trivial to answer. We then introduce a method we call “decision thresholds” for answering it which can be applied to any variation of selection-based DNA computing. Furthermore, we provide an example by applying this method to the technique of using a network of microreactors employing negative selection of ssDNA.

I. BACKGROUND

A. The Boolean Satisfiability problem (SAT)

The Boolean Satisfiability problem (abbreviated SAT) can be simply described as the problem of deciding whether a given boolean expression has any (correct) solutions; more precisely, it can be stated as the following question: Given a boolean expression, do there exist any assignments of truth values (TRUE and FALSE) to the variables of the expression such that the expression as a whole evaluates to TRUE? SAT is a member of an important class of problems known as NP-Complete; it is a property of this class that an efficient algorithm to one of its problems yields an efficient (polynomial) algorithm to all the rest of its problems as well. Unfortunately, no polynomial-time algorithms are currently known for NP-Complete problems. However, SAT (like all NP-Complete problems) is very amenable to high degrees of parallelism, which makes it an ideal candidate for computation using techniques which are massively parallel by nature, such as DNA computing. For these reasons SAT problems provide a natural testing ground for DNA computing research [4], [6], [8].

B. Using DNA computing to compute a boolean expression

A “potential solution” to a boolean expression is an assignment of truth values to the variables of the expression. If a sequence of nucleotides (which we will refer to as a “word”) is used to represent the assignment of a truth value to a variable (e.g. GTTACGTGAGC could mean “A=TRUE”), then a molecule of single-stranded DNA (abbreviated as “ssDNA”) consisting of these words can represent a potential solution to a boolean expression. This is the essence of the DNA computation model, and its attractiveness is in the potential for taking advantage of the inherently parallel nature of biochemical operations to achieve efficient computations for problems that are currently computationally infeasible.

- A library of ssDNA can be built based upon these nucleotide words such that every potential solution to the boolean expression is in the library. Computation of that expression then means determining which potential solutions are correct solutions (solutions such that the value of the expression as a whole is TRUE). In the context of DNA computing, computation of a boolean expression generally refers to some process which begins with a full library of potential solutions, performs biochemical operations on the library, and ideally results in a set of ssDNA containing only those molecules which represent correct solutions. Realistically, this sort of computation will not be perfect, and will leave a (hopefully small) amount of ssDNA which correspond to incorrect solutions [11]. It is this error-prone aspect of DNA computing which must be overcome in order to have a technique which can give confident answers to problems such as SAT.

- There are many variations of the DNA computing model just described, depending on how a boolean expression is translated into a DNA computation and depending on the particular biochemical operations used. This paper addresses selection-based techniques (sometimes also called “extract”, “filtering”, or “sorting” techniques), which attempt primarily to separate correct strands from incorrect strands and then discard the incorrect strands. Although we will be paying particular attention to a technique which uses negative selection, without further examination it cannot safely be assumed that the issues raised in this paper do not also apply to other techniques, including those which rely on positive selection.

II. THE QUESTION: DOES THE EXPRESSION HAVE ANY CORRECT SOLUTIONS?

On the surface, it may seem that the computation of a boolean expression as described above will yield a clear answer to the SAT problem for that same expression. However, it is a surprisingly non-trivial task from a general point of view. The result of a DNA computation is a collection of ssDNA molecules; we must assume there will be some amount of ssDNA in this result which corresponds to incorrect solutions to the boolean expression. Therefore, the task is not to determine whether any ssDNA is present, or even which potential solutions are present, but rather whether any correct ssDNA is present in addition to the incorrect ssDNA.
A. An intuitive idea: data clustering

One possible approach to this task is based on the intuition that, if correct ssDNA is indeed present, it should have orders of magnitude more strands present than incorrect ssDNA. Perhaps, then, an examination of the approximate number of strands remaining for each possible solution would allow us to distinguish whether there are two “groups” of ssDNA present (correct and incorrect), or only one group (incorrect). Thus it appears the task of deciding the answer to an SAT problem based on a DNA computation could be approached as a data clustering problem.

Unfortunately, there are many barriers to using this approach. To begin, such an approach requires measuring the approximate amount of ssDNA for each possible solution. For problems of such magnitude that they cannot be quickly solved by conventional computers, the number of possible solutions can be astronomical. Performing a clustering algorithm on that many data points is likely to be at least as slow as solving the original problem using conventional techniques. Furthermore, clustering algorithms typically focus on the task of deciding which data points are similar enough to belong to the same group, rather than on deciding how many groups would be the most appropriate. Lastly, there is the problem of interpretation of the result. It is possible that, for some boolean expressions, two distinct groups of incorrect ssDNA might be recognized, yielding a clustering result of “two groups” even when there is no correct ssDNA present. Ensuring this does not occur would require not just that incorrect strands be kept to a small amount in the result, but that the DNA computation be engineered so that all incorrect strands will occur with approximately the same frequency in the result. An even worse possibility would be that the incorrect ssDNA and the correct ssDNA have approximately similar amounts of strands remaining in the result, which could lead to clustering result of “one group” even though both incorrect and correct ssDNA are present; this possibility would violate the intuitive assumption that correct ssDNA will always vastly outnumber incorrect ssDNA in the result. If this occurs, then no algorithm might be capable of reliably distinguishing correct ssDNA from incorrect ssDNA by examining only the result of the DNA computation.

B. A case that always breaks data clustering

It turns out this devastating occurrence is indeed possible – correct ssDNA can be present in the result of a DNA computation with approximately the same frequency as incorrect ssDNA, or even with a far smaller frequency. To demonstrate this possibility, it is necessary to first explain the DNA computation model associated with the example pathological case. (While this particular example may or may not have analogs in other DNA computation models, it cannot be safely assumed that other pathological cases do not exist in other models unless further research leads to such a conclusion.)

1) Overview of the negative selection microreactor model: The model we will use in this paper is a microfluidic network of microreactors employing negative selection. The function of a microreactor is to divide the ssDNA into two sets, based on their value assignments for a single variable in the boolean expression (e.g. to separate strands with the word meaning “A==TRUE” from those with the word meaning “A==FALSE”). In theory, both groups could then be retained for use in different parts of the calculation, but usually this is not the case – a microreactor is used to select one of the sets of strands. For negative selection, a microreactor binds and retains the unwanted set of ssDNA, while the remaining ssDNA flows out of the microreactor for further use in the computation. In contrast, positive selection would require binding the ssDNA which would be used for further computation after a “melting” step, while the unbound ssDNA would be discarded [6]. (For more details on the model used in this paper, see [5]. For details of a model which relies on positive selection, see [2].)

Within a microfluidic network, each microreactor represents a variable in the boolean expression. Operators are represented in the structure of the fluids network [7]. An or operator is represented by two reactors in parallel, while an and operator is represented by two reactors in series. In this manner a boolean expression can be translated into a microfluidic structure. For example, the expression \((A \text{ OR } B) \text{ AND } (C \text{ OR } D)\) can be translated as shown in Figure 1(a).

It is trivial to implement NOT on a particular variable simply by hybridizing the opposite set of strands, however a general NOT operation (applied to any boolean expression rather than only a single variable) cannot be implemented in the same straightforward manner as AND and OR. This shortcoming can be avoided by employing de Morgan’s Law to normalize boolean expressions [1], [2]. A microreactor network realization of a more complex boolean expression which utilizes NOT normalization is shown in Figure 1(b).

2) An example pathological case: Let us start with the expression \(A \text{ AND } B\). There are four potential solutions to this expression, listed below:

1) \(A==TRUE, B==TRUE\)
2) \(A==TRUE, B==FALSE\)
3) \(A==FALSE, B==TRUE\)
4) \(A==FALSE, B==FALSE\)

Of these, only the first is a correct solution. Let us also examine the expression \(A \text{ AND } \neg A \text{ AND } \neg B\); this expression has the same four potential solutions, but has no correct solutions (i.e., it is a contradiction). Now, let us augment the first expression by joining it to the second with an OR, to get \((A \text{ AND } B) \text{ OR } (A \text{ AND } \neg A \text{ AND } \neg B)\) (depicted as a microreactor network in Figure 2(a)).

It is straightforward to see that this expression is logically equivalent to the first expression. In fact, we can perform this operation repeatedly (augmenting the expression by joining it to the second expression with an OR) and it will always be logically equivalent to the first expression.

However, from the perspective of a microreactor network this is not at all the same expression. Every time the expression is augmented in this way, the critical branch of
To better understand the impact of these augmentations, it is necessary to closely examine the computation of the subexpression used in each augmentation.

As noted earlier, the subexpression \( A \land (\neg A) \land (\neg B) \) has no correct solutions. However, not all potential solutions will be bound to the same degree by this subexpression. We will examine each of the four potential solutions in turn.

1) \( A = \text{TRUE}, B = \text{TRUE} \): Strands of this form will be filtered twice in the subexpression – once by the microreactor representing \( \neg A \), and once by the microreactor representing \( \neg B \).

2) \( A = \text{TRUE}, B = \text{FALSE} \): Strands of this form will be filtered only once in the subexpression, by the microreactor representing \( \neg A \).

3) \( A = \text{FALSE}, B = \text{TRUE} \): Strands of this form will be filtered twice in the subexpression – once by the microreactor representing \( A \), and once by the microreactor representing \( \neg B \).

4) \( A = \text{FALSE}, B = \text{FALSE} \): Strands of this form will be filtered only once in the subexpression, by the microreactor representing \( A \).

The key point to note is that the first potential solution (which is the only correct solution, from the perspective of the entire expression) will be filtered twice, compared with only once for each of two different incorrect solutions. From this it is straightforward to see that this subexpression will allow orders of magnitude more incorrect strands to pass than correct strands (this effect can be verified using the simulation tool presented in [3]).

With this setup, no matter how many times the augmenting operation is performed, there will always be a contradictory subexpression at the topmost level of ORs, and that subexpression will always pass the same amount and blend of ssDNA (which, as noted above, will contain far more incorrect strands than correct strands); thus, beyond the first few augmentation operations, additional augmentations will not significantly change the amount of incorrect ssDNA in the final DNA computation result. However, each time the augmentation operation is performed, the amount of ssDNA
flowing through the critical branch of computation is halved, thus reducing the amount of correct ssDNA passed by this subexpression by approximately 50% (it is not precisely 50% due to non-linearity in the binding process of ssDNA). The result is that the amount of correct ssDNA passed by this subexpression, \( c \), can be approximated by

\[
c = \frac{1}{2} \times t
\]

where \( k \) is the number of augmentations performed and \( t \) is the total original amount of ssDNA; this value can be made arbitrarily small by increasing the value of \( k \). Regardless of the exact parameters of the DNA computation, by performing this augmentation operation enough times it will always be possible to create an expression that will yield an ambiguous result (one with more incorrect ssDNA than correct ssDNA).

Of course, this situation is contrived, and thus it may seem unlikely that it would occur in a real-world SAT problem. Having shown that it is in any way possible to create this situation, however contrived, we must accept the possibility that similar results will occur in real-world SAT problems unless proven otherwise. Given this inherent unreliability, it may seem that DNA computation is not feasible for solving SAT problems. However, that is not necessarily the case; one merely needs a way to rigorously and efficiently determine whether a given DNA computation will be reliable or not. Any practical technique for answering the SAT question based on a DNA computation must therefore be able to give one of three responses: “Yes, the expression has correct solutions.”, “No, the expression has no correct solutions.”, or “Unable to determine based on this computation.” In the next section, we present one such technique.

III. ONE WAY TO FIND AN ANSWER: DECISION THRESHOLDS

A. General technique overview

The idea for the “decision threshold” technique derived initially from an attempt to recognize the possibility of an ambiguous result such as described above. Stated simply, an “ambiguous result” occurs when the amount of incorrect ssDNA in the result of a DNA computation is at least as great as the amount of correct ssDNA in the result of a DNA computation. Therefore, one way to detect when an ambiguous result is possible is to calculate and compare a pair of values: the least amount of ssDNA which could
(a) Ideally, there will be a gap between the possible value ranges for the resulting ssDNA amount.

(b) Since we cannot know the actual possible value ranges, we must calculate bounds on the ranges. Ideally, there will also be a gap between the weaker bounded ranges.

result if the expression does have correct solutions, and the maximum amount of ssDNA which could result if the expression does not have correct solutions. If the first value is greater than the second value, an ambiguous result cannot occur, as shown in Figure 3(a).

Since exact values may be as inefficient to calculate as solving the initial SAT problem, in practice we must rely on weaker bounds that are simpler to compute than the exact values. For example, rather than computing the least amount of ssDNA which could result if the expression does have correct solutions, we can compute any lower bound on that value (see Figure 3(b)). Using weaker bounds may lead to a false positive (calculating that an ambiguous result is possible when in fact it isn’t), but will not risk a false negative (calculating that an ambiguous result is not possible when in fact it is). (See Figure 4(a).) The closer the bounds are to the exact values, the less risk there is of a false positive.

These bounds, once calculated, can also be used as thresholds to make an SAT decision following a DNA computation, as shown in Figure 4(b). If the amount of ssDNA in the result of the DNA computation is greater than the upper bound on the amount of ssDNA which could result if the expression does have correct solutions, then (if this bound was accurately calculated) we can be certain that the expression does have correct solutions. Similarly, if the amount of ssDNA in the result of the DNA computation is less than the lower bound on the amount of ssDNA which could result if the expression does have correct solutions, then we can be certain that the expression does not have correct solutions. This is true even if there is the possibility of an ambiguous result – see Figure 4(b). If the bounds are accurately calculated, it is impossible for both conditions to be true. If neither of these conditions is true, then we do not know whether the expression has correct solutions.

B. Application to the negative selection microreactor model

As a concrete example of the determination of these decision thresholds, we will show a way to determine decision thresholds for the negative selection microreactor model; we will also show example calculations based on them.

1) Lower bound of correct ssDNA: First, we will find a lower bound for the amount of ssDNA which could result if the expression has correct solutions. In this DNA computing model, a potential solution is a correct solution if and only if there exists a path through the microreactor network which...
does not filter that solution. In the worst case, there would be only one correct solution, and only one such path would exist for that solution. Since we are looking for a lower bound, it is safe to assume that no correct ssDNA in the result comes through any other path. To determine the minimum amount of correct ssDNA which could pass through a single path without being filtered, we will need to use information about the behavior of individual microreactors. In particular, we will need to be able to estimate how much correct ssDNA is lost in each microreactor. For this model, no non-specific binding takes place in microreactors (e.g. no ssDNA is lost in microreactors except that which is being filtered), so we can assume all correct ssDNA which enters this path will pass through into the result; see the conclusion for further discussion on this. All that is left, then, is to determine the minimal amount of correct ssDNA which can travel through a single path. This can be accomplished via an examination of the boolean expression; a simple recursive algorithm can determine this value in time proportional to the number of operators in the expression. The lower bound on correct ssDNA, \( L \), can be calculated as:

\[
L = m \ast \frac{1}{2^x} \ast t
\]

where \( m \) is the minimal amount of correct ssDNA which can travel through a single path, as a proportion to the total amount of ssDNA entered into the network, and \( x \) is the number of unique variables in the boolean expression.

2) Example of calculating a correct ssDNA lower bound: Consider the expression \(((A \text{ AND } (\text{NOT } C) \text{ OR } D)) \text{ OR } (B \text{ AND } C \text{ AND } D) \text{ OR } (\text{NOT } (F \text{ AND } G))) \text{ AND NOT } H\), depicted in Figure 1(b); the highlighted path shows a minimal-strength path through the network – \( \frac{1}{16} \)th of the total ssDNA will travel through that path. \( \frac{1}{16} \ast \frac{1}{2^6} \ast t = \frac{1}{16} \ast \frac{1}{64} \ast t = \frac{1}{1024} \ast t \), so a lower bound of correct ssDNA calculated as described in the previous section would be \( \frac{1}{16} \) of the total initial ssDNA amount. This is a severe underestimate for this particular expression, and more accurate bounds are desirable and likely possible, but nevertheless it is still a correct lower bound.

3) Upper bound of incorrect ssDNA: Now, we will find an upper bound for the amount of ssDNA which could result if the expression does not have correct solutions. In this DNA computing model, a potential solution is an incorrect solution if and only if there exists no path through the microreactor network which does not filter that solution; in other words, a potential solution is an incorrect solution if and only if every
path through the microreactor network will filter that solution at least once. In the worst case, every potential solution is an incorrect solution, and each solution is filtered only once. The amount of incorrect ssDNA which passes through a microreactor is heavily dependent on several parameters (see [5]); therefore, calculating an accurate upper bound depends on simulation of a microreactor. Unfortunately, simulation of a full microreactor network requires an impractical amount of both computation and memory for sufficiently large instances of the SAT problem, so we must find a way to calculate an upper bound using very little simulation. In the case of this model, the more incorrect ssDNA passes through a given microreactor relative to its binding capacity, the greater the proportion of incorrect ssDNA that will be allowed to pass through the microreactor. An upper bound on incorrect ssDNA, \( U \), can therefore be determined by simulating a single microreactor with an amount of ssDNA flowing through it equal to the entire network's initial ssDNA amount:

\[
U = s
\]

where \( s \) is the proportion of ssDNA is incorrectly unfiltered in a single microreactor simulation as described above; it is assumed to be the amount of incorrect ssDNA which passes through the entire network. Although simulation of an entire network is a prohibitively costly operation, in this scheme it is never necessary to simulate more than a single microreactor, which is an acceptably small (and constant) cost. However, this scheme does not take into account the structure of the microreactor network; doing so could likely give a more accurate, if more costly, upper bound.

\textbf{IV. Conclusion}

By showing the form of a contradicting example, we have shown that it is not sufficient to assume that calculation of a boolean expression using DNA computing techniques will yield a clear answer to the corresponding SAT problem. However, we have also shown that this contradicting example does not invalidate DNA computing as a method for solving SAT problems. For any DNA computing technique purported to answer an instance of the SAT problem, it is necessary either to prove that such cases cannot occur for the technique, or to provide a method by which one can confidently answer the SAT question based on the results of a DNA computation (or decide that a confident answer cannot be given). We have provided the “decision thresholds” method as one way of doing this, and applied it to the DNA computing technique examined in our prior research, which is based on negative selection in a microreactor network. Additionally, this method can be used before running a DNA computation in order to warn of potentially ambiguous results, allowing the researcher to work to mitigate this possibility before running the time-consuming DNA computation.

There are two very important implications in these findings. First is the potential importance of algorithms and heuristics for increasing the correct ssDNA which passes through a network and for decreasing the incorrect ssDNA which passes through a network. These can use boolean algebra to change the boolean expression into an equivalent one more suited to DNA computation, or they could help determine appropriate parameters for the DNA computation itself. By increasing the correct ssDNA or decreasing the incorrect ssDNA, one would reduce the likelihood of wasting a DNA computation (by producing an ambiguous result that cannot confidently yield an answer to the SAT question). Furthermore, the research involved in developing such algorithms and heuristics would likely lead to a better understanding of the properties of DNA computing, which could in turn lead to the development of stronger decision thresholds.

The other important implication is that developing simulations of DNA computations can be useful not just for aiding research, but also to assist those who will be using DNA computation to solve real problems. For example, simulation of a single microreactor is an important step in the calculation of the decision thresholds presented in this paper. The accuracy of such simulations are crucial to the reliability of the decision thresholds calculated from them. It was noted that the simulation model used for this paper assumes no non-specific binding takes place (i.e. microreactors do not bind any ssDNA they do not intend to bind). In reality, this is not strictly true. Developing a simulation model which takes this factor into account would decrease the chances of incorrectly deciding that an ambiguous result is not possible for a given DNA computation, or (even worse) mischaracterizing an ambiguous result as decisive, thus possibly giving an incorrect SAT decision. Such a simulation model would complement the two additional approaches of developing error-correcting implementations [11] and more robust computational/biochemical techniques [12], which in turn might be used in conjunction with decision thresholds to reduce the opportunities for false positives.

\textbf{V. Acknowledgements}

\textit{J. W. Jaromczyk acknowledges the support by the Kentucky Biomedical Research Infrastructure Network, funded by grant 2P20RR016481-04 from the National Center for Research Resources and awarded to Nigel Cooper from the University of Louisville.}

\textit{D. van Noort acknowledges that a portion of the work on this paper took place during a visit to the University of Kentucky.}

\textbf{References}


