Whiplash PCR for $O(1)$ Computing

Erik Winfree
California Institute of Technology
winfree@hope.caltech.edu

Caltech Technical Report CS-TR:1998.23
May 31, 1998

Abstract This paper reviews the experimental technique of *whiplash PCR*, as introduced in Hagiya et al. (in press), and proposes a model of computation based on this technique in combination with *assembly PCR* (Stemmer et al. 1995). In this model, based on *GOTO graphs*, a number of NP-complete problems can be solved in $O(1)$ biosteps, including branching program satisfiability, the independent set problem, and the Hamiltonian path problem. In addition, we propose a simple extension of the experimental technique that allows single DNA strands to simulate the execution of a feed-forward circuit, giving rise to a solution to the circuit satisfiability problem in $O(1)$ biosteps.

1 Introduction

In an ingenious paper, Hagiya et al. (in press) introduce an experimental technique they call *polymerization stop* and theoretically show how by thermal cycling, individual DNA molecules can compute the output of Boolean μ-formulas (and-or-not formulas in which every variable is referenced at most once). Because each DNA molecule repetitively forms hairpins so that it can serve simultaneously as both "primer" and "template" for a stopped polymerase reaction, Adleman has dubbed this experimental technique *whiplash PCR*. Hagiya et al. (in press) describe how whiplash PCR can be used to solve the problem of learning μ-formulas given positive and negative data, and more recently Sakamoto et al. (in press) has shown how other NP-complete problems can be solved with whiplash PCR[1].

The motivation for whiplash PCR begins with the interpretation of DNA polymerase as an enzymatic Turing Machine implementing the simply COPY operation. Bennett (1982) goes farther and imagines designing a set of enzymes to simulate the operation of an arbitrary Turing Machine, but these ideas were never implemented because of the difficulty of designing enzymes *de novo*. But is the existing polymerase enzyme's computational capability limited to just copying? Recently, Leete et al. (in press) realized that the hybridization of primers

[1]Sakamoto et al. (in press) use the term *successive localized polymerization* to allow for the possibility of intermolecular reactions as well as intramolecular reactions.

in the polymerase chain reaction (PCR) provides information-based control over the COPY operation, and that complex computations (such as the symbolic expansion of determinants) can be carried out in DNA using a series of PCR reactions. However, this is a very labor-intensive series of laboratory procedures, and it has not yet been attempted experimentally. Hagiya et al. (in press) adds two key insights: (1) that polymerase copying activity (which was initiated by the primer sequence) can be conveniently terminated by a "stop sequence" in the template DNA; and (2) that if the $3'$ end of a DNA strand serves as the same strand's primer, then an individual DNA molecule can be a self-contained computational unit. It was shown how in a single reaction, each DNA strand can independently compute the result of a μ-formula, and how the problem of learning μ-formulas from N positive and negative examples can be solved in in $O(N)$ biosteps. (We use the term "biostep" to refer to a single laboratory procedure. Many chemical reaction steps can take place during a single biostep; in whiplash PCR, the many chemical reactions are sequenced by thermal cycling.)

The DNA used in whiplash PCR has the form $5'$-$\mathbf{stop}_1$-$\mathbf{new}_1$-$\mathbf{old}_1$- $\cdots$ -$\mathbf{stop}_n$-$\mathbf{new}_n$-$\mathbf{old}_n$-**head**-$3'$. When the $3'$ end (head) of the DNA strand anneals to a DNA sequence $\mathbf{old}_i$, polymerase copies the sequence $\mathbf{new}_i$, and the polymerase is stopped and dissociates upon encountering the sequence **stop** (for example, because the stop sequence is GGG and the polymerase buffer contains only A, T, and G). The head of the DNA now contains a new sequence. Upon the next thermal cycle, the head can anneal to a different **old** location, and copy the corresponding **new** sequence. We will refer to the basic DNA unit $5'$-**stop-new-old**-$3'$ as a *frame* and use the notation (**new old**). In general, **boldface** will be used when referring to DNA sequences, while *italics* will be used when referring to logical variables.

We describe by example the method given in Hagiya et al. (in press) by which a single DNA strand computes a μ-formulas during whiplash PCR. Consider the μ-formula $f = (x_1 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4)$. This can be translated to the decision process shown in Figure 1, wherein variable x_1 is checked first; if it is false (written False, 0, or $-$) then variable x_3 is checked, etc. Decision processes of this form are known as *branching programs*[2]; they have already arisen in the study of DNA computing based on affinity separation (Winfree 1996). Here we have the restriction that each variable be accessed at most once; we call these μ-branching programs. μ-branching programs can represent more functions than μ-formulas; in the absence of this restriction, branching programs are provably more concise than formulas[3].

The translation of an n-variable μ-branching program into DNA makes use of the $3n+2$ DNA sequences $\{\mathbf{x}_1, \mathbf{x}_1^-, \mathbf{x}_1^+, \cdots, \mathbf{x}_4^+, \mathbf{out}^-, \mathbf{out}^+\}$. Each edge in the diagram, say the $-$ edge from node i to node j, is then converted into a DNA frame $(\mathbf{x}_j\ \mathbf{x}_i^-)$, which may be read as "if x_i is False, check x_j next." A recursive formula is given in Hagiya et al. (in press) that converts any μ-formula directly into a sequence of DNA frames, the *program frames.* To tell the DNA the values of the input variables, we use additional frames of the form $(\mathbf{x}_i^+\ \mathbf{x}_i)$, read as "$x_i$ has the value True;" these are the *data frames.* The data frames and the program frames are concatenated into a single strand of DNA, with an initial $3'$ head sequence complementary to $\mathbf{x}_1$. Figure 2 gives a full set of frames used to implement f and shows how the computation

[2] Also known as *binary decision diagrams.*

[3] For example, the best known procedure for finding and-or-not formulas implementing symmetric functions results in formulas of size $O(n^{4.37})$, whereas branching programs of size $O(\frac{n^2}{\log n})$ can be achieved.

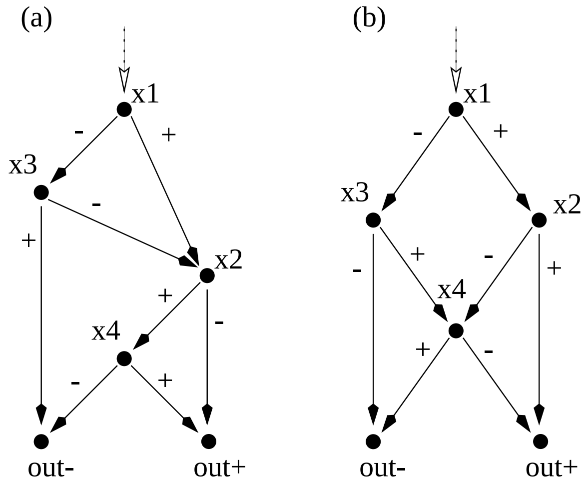

Figure 1: (a) A branching program for computing the μ-formula $(x_1 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4)$. A possible input would be $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1$, which leads to output $^+$. The computation follows a path through the diagram, and thus can only access variables in the order prescribed. (b) A branching program which does not correspond to a μ-formula.

proceeds during whiplash PCR: the head initially anneals to the data region to read the value of x_1; in the next thermal cycle, the head anneals to the frame representing the appropriate edge out of node 1 in the program region, to determine which variable must be checked next; in the next cycle, the head anneals again to the data region, and so on[4]. Because the head might anneal to its previous location (in which case the polymerase is immediately dislodged by the **stop** sequence and nothing happens), the computation proceeds at approximately 1 logical step per two thermocycles. In this fashion, every DNA strand computes in parallel, each containing its own data and its own program.

In the inductive inference problem discussed in Hagiya et al. (in press), one starts with a combinatorial library of DNA representing all μ-formulas of a given size. In each iteration, a positive or negative input example is evaluated by each DNA strand: DNA representing the input is ligated to all remaining DNA strands, which are then evaluated in parallel using whiplash PCR. Those DNA strands computing the correct output value are retained, and the program region is cut from the data and head regions in preparation for the next round of the iteration. After all input examples have been processed, the only DNA programs that remain represent μ-formulas which agree with all examples, and the inductive inference problem has been solved in $O(N)$ biosteps.

By starting with a combinatorial library of DNA representing possible inputs, Sakamoto et al. (in press) describe how whiplash PCR can also be used to solve other NP-complete problems, including conjunctive-normal-form satisfiability (CNF-SAT), Vertex Cover, Direct Sum Cover, and Hamiltonian Path. In the next two sections, we develop similar results for general formula

[4]The restriction that each variable be used at most once arises because the value of the variable itself, encoded in DNA as $\mathbf{x}_i^{\pm}$, is used to keep track of where the computation is in the decision diagram; if there were two nodes which check variable i, then the computation could return to the wrong place in the diagram because there would be two frames matching $\mathbf{x}_i^{\pm}$.

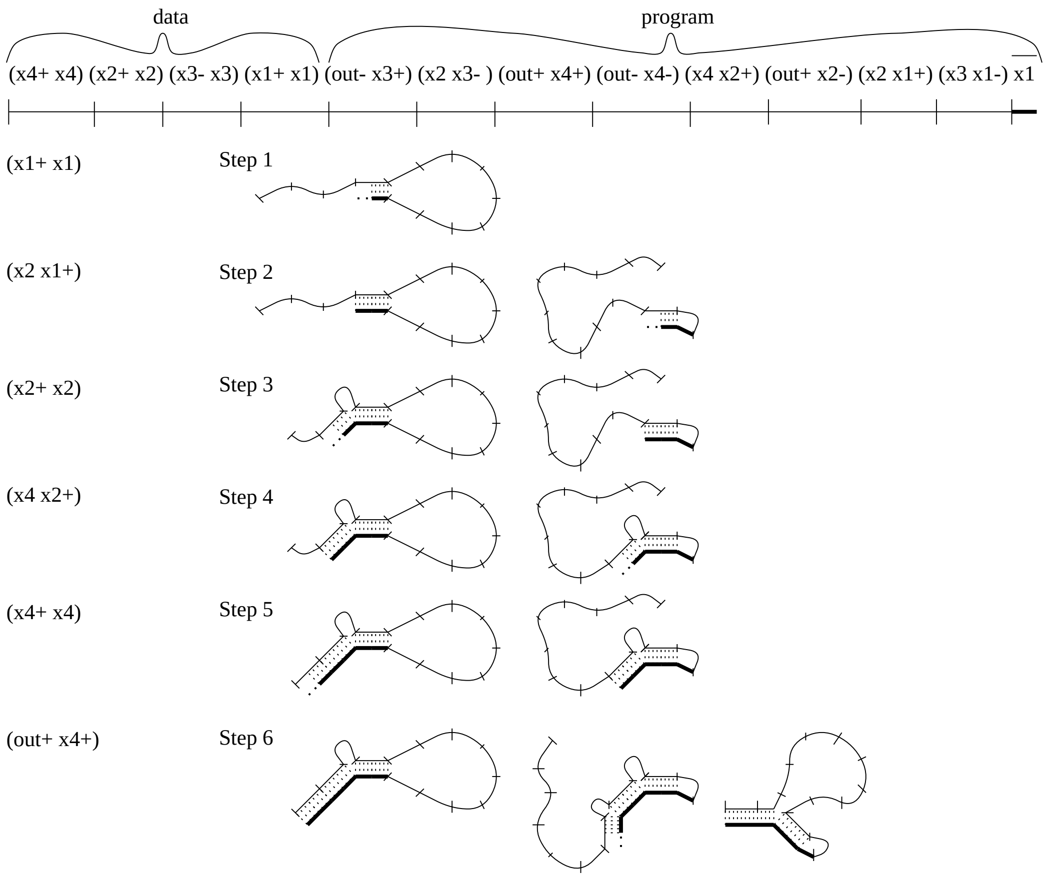

Figure 2: Probable secondary structures during the computation of the μ-formula $(x_1 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4)$ on the input 1101. "Probable" is in the mind of the artist. Note that the tick marks denote the **stop** sequence; because the $3'$ head sequence will never contain the complement to the **stop** sequence, this will be the site of a small bulge in regions that are shown as double-stranded.

satisfiability (FSAT), branching program satisfiability (BP-SAT), Independent Set, and Hamiltonian Path. We suggest the *assembly graph* formalism for the assembly PCR technique, and the *GOTO graph* formalism for describing computations possible by performing assembly PCR and whiplash PCR followed by a single affinity separation.

2 Solving FSAT in $O(1)$ biosteps

Even though a single strand of DNA can only compute the result of a μ-formula, it is possible to solve the formula satisfiability problem in $O(1)$ biosteps – without the restriction that each variable can occur at most once.

Consider the Boolean formula $f = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_3)$. It is a function of $n = 3$ variables, and it accesses one of them more than once; thus it is not a μ-formula. However, if we introduce the new variables $x_{11} = x_{12} = x_1$, then the same function is computed by the μ-formula $\hat{f} = (x_{11} \vee \overline{x_2}) \wedge (\overline{x_{12}} \vee x_3)$, with the additional constraint that $x_{11} = x_{12}$.

In general, if f is a Boolean formula in n variables in which variable i is accessed σ_i times, then we can construct a μ-formula $\hat{f}$ in $\hat{n} = \sum_{i=1}^{n} \sigma_i$ variables, which computes the identical function for input which is appropriately constrained. Specifically, for each $1 \leq i \leq n$, we require $x_{i1} = \ldots = x_{i\sigma_i}$.

We can use the biochemistry of whiplash PCR to compute the μ-formula, and use the biochemistry of hybridization to generate a combinatorial library of DNA representing all possible inputs which obey the equality constraints. Following Adleman (1994), the combinatorial library consists of DNA representing paths through a graph. We use bipartite *assembly graphs*, in which nodes are either black or white and are labelled by distinct single symbols, and directed edges are labelled by symbol strings (possibly length zero) whose symbols are disjoint from those used at nodes. Each symbol represents a unique sequence of DNA. An oligo is generated for each edge in the graph, using the sequences for the symbols of the origin node, the edge, and the destination node: since the graph is bipartite, edges are either from white nodes to black nodes (in which case "sense" oligos are synthesized), or from black nodes to white nodes (in which case the Watson-Crick complementary "anti-sense" oligos are synthesized). These oligos may be mixed in a single test tube and full-length product may be generated using *assembly PCR*[5] (Stemmer et al. 1995). This reaction creates long "repetitive" DNA, which may then be cut at a restriction site to yield defined-length product, and then made single-stranded. For each path through the graph, the sequence of node and edge symbols on that path will be generated in DNA by assembly PCR; the complementary DNA will also be generated[6]. Figure 3 gives an assembly graph for generating all DNA representing inputs where $x_{11} = x_{12}$.

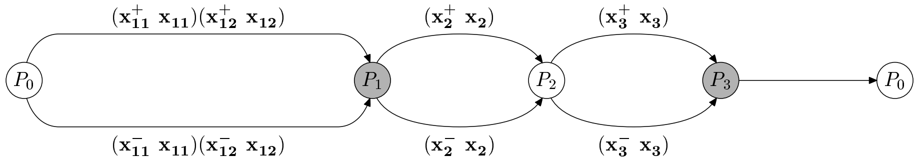

Figure 3: An assembly graph for generating input to the formula $(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_3)$. Up to $2n+1$ oligos are required, and additional symbols P_i are used. For convenience, the node P_0 is written twice. Since there will be a restriction site in $\mathbf{P}_0$, this results effectively in paths from the leftmost node to the rightmost.

Thus, for any μ-formula $\hat{f}$, we can generate a combinatorial library of DNA representing all possible inputs satisfying the equality constraints $\{x_{i1} = \ldots = x_{i\sigma_i}\}$. After assembly of the input DNA, DNA representing $\hat{f}$ can be ligated to the end of all input DNA, the whiplash PCR reaction performed, and DNA whose $3'$ end is $\mathbf{out}^{+}$ extracted. This DNA contains the

[5] This technique is preferred over annealing and ligation due to its improved yield and accuracy; it was used in Ouyang et al. (1997) to create a full library of 6-bit inputs. Note that if the oligos are simply annealed, there are gaps in the double-stranded DNA; these gaps are filled in by the polymerase during assembly PCR. If, as in Adleman (1994), ligation rather than assembly PCR is preferred, then additional oligos must be generated complementary to the frames on the "anti-sense" strands. Of course, for either ligation or assembly PCR to be effective, careful design of the oligos is required; see, for example Deaton et al. (in press).

[6] To be assembled by ligation, no gaps may be present in the the "sense" strand; therefore all "anti-sense" edges must be labelled by the empty string, or additional oligos complementary to the single-stranded "anti-sense" regions must be synthesized. A general assembly graph can be easily transformed into one suitable for ligation by either of these two modifications.

input which satisfies the original formula f. We have solved FSAT in $O(1)$ biosteps (granting that the number of thermocycles necessarily will scale with the size of the formula). The exact procedure described above can also be used for the slightly more difficult BP-SAT problem.

3 Combinatorial Sets of GOTO Programs

We would now like to generalize the techniques used to solve FSAT. To solve FSAT, a sequence of three laboratory procedures was employed: combinatorial generation of DNA by assembly PCR, evaluation of μ-formulas by whiplash PCR, and selection of DNA evaluating to True by affinity separation. Here we introduce a new formalism to describe the computations which can be performed in this manner; this formalism suggests several optimizations and new applications of whiplash PCR.

Our interest comes from the following simple observation: On a given strand of properly constructed DNA, whiplash PCR can be considered as executing a BASIC program consisting entirely of GOTO statements: e.g. the DNA frame $(\mathbf{x}_j\ \mathbf{x}_i)$ can be thought of as "Line i: GOTO line j", or just $i \to j$. The special "line numbers" are $START = 1$, $ACCEPT = \mathbf{out}^+$ and $REJECT = \mathbf{out}^-$. The sequential order in which the GOTO statements appears does not matter, but no line number may appear on the left hand side twice. By using combinatorial synthesis to create a huge number of different programs, and extracting the accepting ones, we are able to solve some interesting mathematical problems. We define a combinatorial set of GOTO programs using a bipartite assembly graph where edges are labelled (possibly with repetition) by GOTO statements and nodes are labelled (uniquely) from P_i. We will insist that all paths generate valid GOTO programs, in which no line number appears twice on the left hand side[7]. This implies, among other things, that the graph has no cycles.

Thus, we consider the following question: Given a graph as defined above, is there a path that generates a GOTO program that reaches $ACCEPT$ when started at line 1? Call this the *GOTO graph satisfaction problem*, or GG-SAT. GG-SAT thus formalizes what can be computed in $O(1)$ biosteps by applying assembly PCR followed by whiplash PCR and affinity separation.

As an example, we will reduce BP-SAT to GG-SAT. Three resource measures of importance are the number of paths through the graph (corresponding to the number of DNA strands generated); the maximal length of the GOTO programs thus generated (corresponding to the length of the DNA strands); and the size, in number of edges, of the GOTO graph (corresponding to the number of DNA oligos that must be synthesized). Then, as shown in Figure 4(a), n-variable m-node BP-SAT can be solved by creating 2^n programs of length $2m + n$, using a GOTO graph of size $2(n+m)$. m lines of the program are fixed; the other m lines are generated in independent blocks of σ_i lines, with two possibilities for each.

This notation makes it obvious that the fixed portion of a GOTO graph is redundant; we can reduce each graph to a smaller one by following all the GOTOs in the fixed portion. The example in Figure 4(a) reduces to just 3 nodes as shown in Figure 4(b). Thus we get the

[7] DNA programs in which a line number appears more than once on the left hand side would execute *probabilistically.*

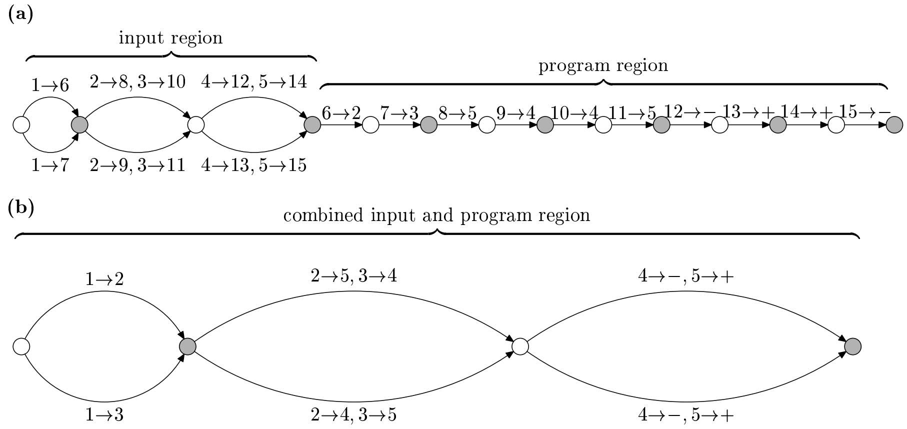

Figure 4: Reducing BP-SAT to GG-SAT: the $n = 3, \hat{n} = 5$ example. (a) The direct construction, combining the assembly graph from Figure 3 and the μ-formula program for $(x_{11} \vee \overline{x_2}) \wedge (\overline{x_{12}} \vee x_3)$. (b) The optimized construction obtained by following GOTO statements in the fixed region of (a). All GOTO programs are of length 5.

improved theorem that n-variable m-node BP-SAT can be solved by creating 2^n programs of length m using a GOTO graph of size $2n$. The m lines are generated in independent blocks of σ_i lines, with two possibilities for each. Because this decreases both the length of the DNA and the number of cycles to complete the program, this construction could be important for experiments solving BP-SAT. It would be interesting to find general polynomial-time algorithms for "optimizing" or "compressing" arbitrary GOTO graphs, in the sense that the new graph solves the same problem but contains fewer paths and/or shorter programs.

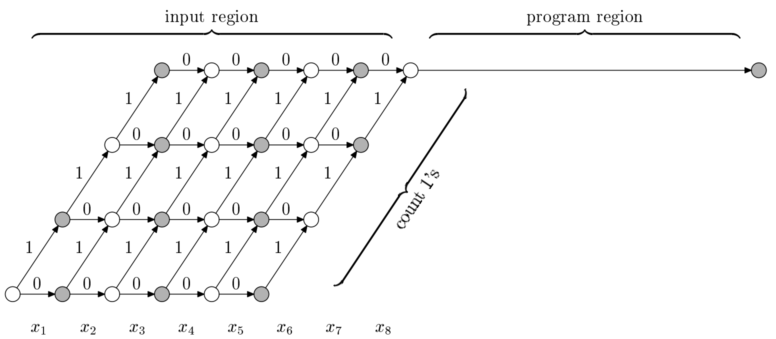

Figure 5: A GOTO graph for solving the Independent Set Problem. Inputs are generated in which exactly $k = 3$ out of $n = 8$ variables have value 1. The edge labels "0" and "1" in column i are shorthand for GOTO statements setting the value of variable x_i; as in FSAT, variables which are referenced more than once in the formula must be duplicated, and the corresponding edges in the graph will be labelled with more than one GOTO statement. Note that concentration ratios of the oligos could be adjusted to make all paths equally likely (for ligation-based assembly, at least; it is not so clear for assembly PCR).

However, we are still failing to fully exploit the expressive power of the graph; so far we have considered only essentially linear graphs. In the context of circuit satisfiability, Boneh et al. (1996) commented that providing a regular language as input to the circuit, rather than just $\{0, 1\}^*$, could for some problems both reduce the size of the circuit and decrease the volume of DNA needed to solve the problem, and that the desired n-bit input can be provided by assembling DNA paths through a graph of size nM, where M is the size of a finite state machine recognizing the regular language. The same comment holds true for BP-SAT. A simple example follows from the ideas in Bach et al. (1996): the polynomial time 2SAT problem becomes NP-complete when given the restriction that satisfying solutions must have exactly k ones. An instance is the Independent Set Problem, which asks, given an undirected graph and an integer k, is there a subset of k vertices which have no edges among themselves? The 2-CNF formula we will use for this problem is

$$\wedge_{s=1}^{e}(\overline{x_{i_s}} \vee \overline{x_{j_s}})$$

where the graph has edges $[i_1, j_1] \ldots [i_e, j_e]$ and x_i indicates membership in the independent set. The formula simply checks that no two chosen vertices have an edge between them. To solve the problem, we ask for a solution to this formula in which *exactly* k variables are 1. This is done in DNA by generating only inputs with k variables set. A GOTO graph for this problem is shown in Figure 5; variables used more than once must be duplicated, and the fixed GOTO statements in the "program region" can be eliminated just as in the BP-SAT optimization.

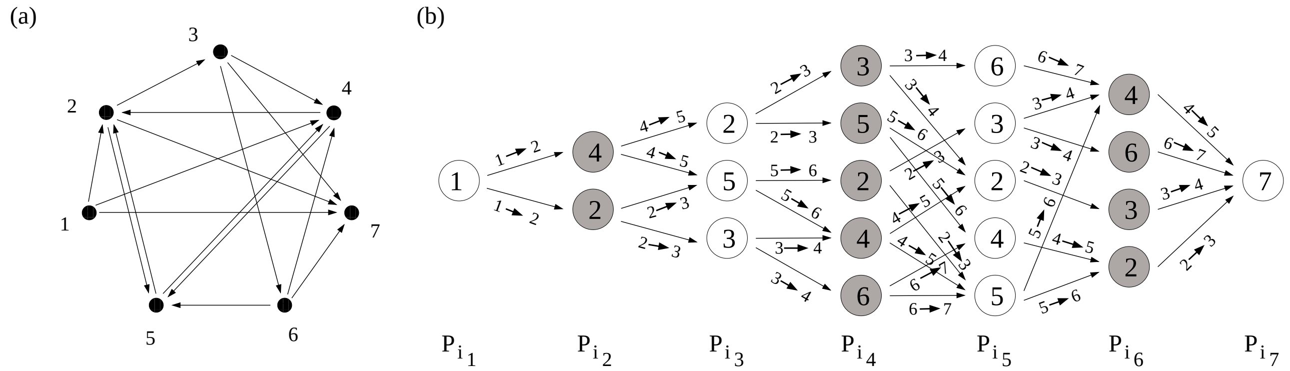

Figure 6: Solving the Hamiltonian Path Problem: A graph G (a) and its corresponding GOTO graph GG (b). This is Adleman's example with 2 additional edges added to prevent pruning from simplifying the GOTO graph to triviality. For convenience the nodes show only the vertex index i, and not the full symbol P_{i_k}.

As a final example, we consider the Hamiltonian Path Problem (HPP) solved in Adleman (1994). Our procedure begins by converting (in polynomial time) the original graph G into a GOTO graph GG. Suppose G has n vertices; then GG will have n^2 vertices, arranged in layers, such that if there is an edge $[i, j]$ in G, then in the GOTO graph, for each $k \in \{2 \cdots n\}$ there is an edge $[P_{i_{k-1}}, P_{j_k}]$, labelled $i \rightarrow (i+1)$ (with $ACCEPT = n$). Since we are only interested in paths from vertex 1 to vertex n, we prune the new graph to include only vertices which may be reached from P_{1_1} and which may reach P_{n_n}; this dynamic programming problem takes time $O(n^2)$ on an electronic computer. We now have the GOTO graph GG, as shown in Figure 6. If G has E edges, then GG requires less than E^2 oligos.

Every path through GG represents a length n path through G from vertex 1 to vertex n. A Hamiltonian path will contain, in some order, the frames

$$\{1 \rightarrow 2, 2 \rightarrow 3, \cdots, (n-1) \rightarrow ACCEPT\},$$

and thus the GOTO program, as executed by whiplash PCR, will proceed to $ACCEPT$. All other paths will duplicate some frame and lack another – these GOTO programs will terminate and never reach $ACCEPT$. Consequently, extraction of DNA containing the $ACCEPT$ sequence will identify the Hamiltonian path, and we have solved HPP in $O(1)$ steps.

4 Single-Strand Computation of Boolean Circuits

Using whiplash PCR in the manner suggested in Hagiya et al. (in press), where exactly one symbol is copied in each polymerization stop step, gives each strand exactly the computational power of a GOTO program, and no more. However, whiplash PCR may give each strand more computational power, if copying more than one symbol is experimentally feasible. The idea is this: when the head of the DNA strand is being extended, it might not only change the "state" of the head but also add a new "program" frame.

Suppose for the moment that the variables x_i are encoded by $\mathbf{x}_i, \mathbf{x}_i^+, \mathbf{x}_i^-$ using A, T, and C, and that the new *gate* variables g_i are encoded by $\mathbf{g}_i, \mathbf{g}_i^+, \mathbf{g}_i^-$ using exclusively A and T. G and C are respectively used for representing the **stop** sequence and its complement. The polymerization buffer still includes A, T, and G, but not C. The restricted alphabet used for the gate symbols makes designing DNA sequences a more difficult task[8], but it is necessary for the construction we give below because now a gate symbol can be copied by polymerase *twice* during whiplash PCR.

In our original discussion of branching programs, a + edge from the node reading x_7 to the node reading x_4 would be encoded by the frame $(\mathbf{x}_4\ \mathbf{x}_7^+)$. During biochemical execution with whiplash PCR, a transition through this edge would entail hairpin formation with binding to $\mathbf{x}_7^+$ and polymerase extension copying $\mathbf{x}_4$, as shown in Figure 7(a). Our new proposal involves copying more than $\mathbf{x}_4$ during the polymerase extension, thereby memorizing an intermediate result of the computation. In Figure 7(b) we show the execution of an *enhanced frame* $(\mathbf{x}_4\ \overline{(\mathbf{g}_8^+\ \mathbf{g}_8)}\ \overline{(\mathbf{g}_5^-\ \mathbf{g}_5)}\ \mathbf{x}_7^+)$. Here, the original DNA encodes for the "anti-sense" of a valid frame, and thus the frame is inactive, or *hidden*. The two hidden frames present here are intended to assign values to new variables g_5 and g_8, but that assignment will not become effective while the frame is still hidden. However, if the enhanced frame is executed, the hidden frames are copied as "sense" frames onto the growing $3'$ end of the DNA, thus activating the hidden frames for potential future use. The final $3'$ sequence of the DNA will still be $\mathbf{x}_4$, which will determine the immediate course of the computation as usual.

At subsequent points in the evaluation, reference can be made to look for the values of g_5 or g_8. These values will be found by the head hybridizing to the newly activated frames and copying to the GGG stop sequence – only now the head will not be hybridizing to the "input" part of the DNA, but to part of the growing "head history" itself.

[8] An expanded DNA alphabet, making use of artificial base pairs which are both highly specific and can be incorporated by DNA polymerase, would allow greater flexibility in sequence design; indeed, Sakamoto et al. (in press) reports preliminary studies of using iso-C and iso-G (Switzer et al. 1993) in whiplash PCR. If this chemistry is successful, the variables x_i and g_i could be encoded using A, T, C, and G; the **stop** sequence could be iso-G-iso-G-iso-G and its complement iso-C-iso-C-iso-C; and the polymerization buffer could contain A, T, C, G, and iso-G.

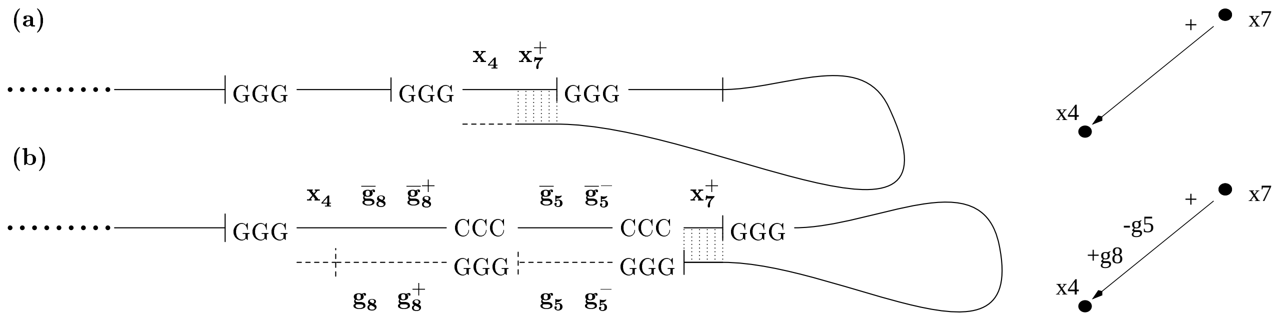

Figure 7: (a) The polymerization stop step on a standard frame, where a single symbol is copied, and its representation as an edge in a BP. (b) The polymerization stop step on an enhanced frame, where two hidden frames are made active, and its representation as an edge in a WOBP.

What is the use of activating hidden frames? The possibility of memorizing intermediate results gives rise to a model of computation that we call *write-once branching programs* (WOBP)[9]. Each node still has two outgoing edges, one labeled $+$ and the other $-$; however, edges may now also have the additional labels $\pm g_i$, which indicate that the variable g_i is to be assigned the value $+$ or $-$. For implementation using whiplash PCR, a restriction is imposed: again, a given variable may be read at most once, and nodes may be labeled to read any input variable x_i or any gate variable g_i, so long as all paths to a given node have assigned exactly one value to the gate variable being read[10]. We call these restricted programs μ-WOBP.

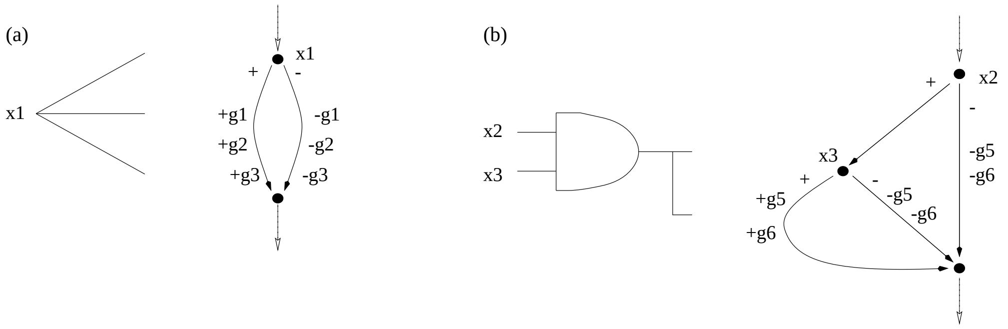

Figure 8: (a) Input variables with multiple fan-out are handled by reading them once, and writing multiple distinct gate variables which may subsequently be read once each. (b) The translation of a gate with fan-out 2 into a write-once branching program requires two decision nodes (only one of which is guaranteed to be used). Two new gate variables are written. To translate an entire circuit, first the input variables and then the gates would be processed in linear order in the branching program. Clearly, much more efficient translations are possible; for example, gates with fan-out 1 need not be memorized.

μ-WOBP are at least as concise as circuits[11]; a circuit with n inputs accessed in total $\hat{n}$ times, and g gates with total gate fan-out p can be implemented in a μ-WOBP using no more than

[9]This model can also be used to describe DNA computation performed by a sequence of affinity separations and ligations, as in Boneh et al. (1996).

[10]Again, we have a probabilistic model if this restriction is violated.

[11]The converse is also true: a circuit can be constructed in which (usually) two gates are used for each edge

$n+2g$ nodes and $\hat{n}+p$ gate variables[12]. The simple construction uses the building blocks shown in Figure 8. First, each input variable x_i is read and duplicated into σ_i new variables, so every subsequent read uses a unique variable. Then, each circuit gate is processed in turn, and its output is stored in a new gate variable (or variables, if the gate has fanout greater than unity). The translation of a small circuit is shown in Figure 9. Thus, we can theoretically solve the circuit-SAT problem in "one pot" using whiplash PCR.

In the case shown in Figure 9, a much smaller μ-WOBP (essentially a BP) exists which computes the same function, pointing out that our construction of a μ-WOBP from a circuit is not the most efficient construction possible. However, for more difficult problems, circuits can be much more efficient than branching programs[13]. This means that a fixed size CSAT problem may be more difficult than a BP-SAT problem of the same size.

One serious concern is that the problem of secondary structure interfering with the progress of the computation is made worse. First, "inopportune" hybridization now involves much longer subsequences, resulting in many thermocycles in which no progress is made. Secondly, newly activated frames are located in the "head history" region of the DNA, which is likely to be involved in secondary structure. Experimental investigation is required to see how serious the problems will be.

5 Conclusions and Future Directions

Like other forms of DNA computation, it seems that whiplash PCR can't by itself compete with electronic circuits unless there are significant advances in the control of the biochemistry. However, the computational power of whiplash PCR – in theory – suggests that "one-pot" biochemical reactions have more potential for computation than previously thought. Conceivably, whiplash PCR could be combined with other kinds of DNA processing – either stepwise or within the "one pot" biochemical reaction. For example, we can consider modifications of whiplash PCR wherein DNA strands not only grow though polymerization, but also *shrink* due to other enzyme activity (e.g. restriction endonucleases or topoisomerases). An open theoretical question is how to use non-determinism during whiplash PCR: we have already discussed the case where the solution to a problem is found by first using nondeterministic steps in the generation of the DNA, and then using deterministic steps during the execution of the program, but whiplash PCR could equally well be used to perform nondeterministic steps by having multiple frames matching the current head state.

in the WOBP to test if the edge was traversed during computation. Thus a circuit with $3m$ gates can be constructed from a WOBP with m nodes.

[12]Just $2g$ nodes and p gate variables are required if we allow preparing the input with duplicated variables, as in the FSAT construction.

[13]As a simple example, an arbitrary symmetric function can be implemented in a circuit of size $O(n)$, but the best construction for branching programs requires $O(\frac{n^2}{\log n})$ nodes.

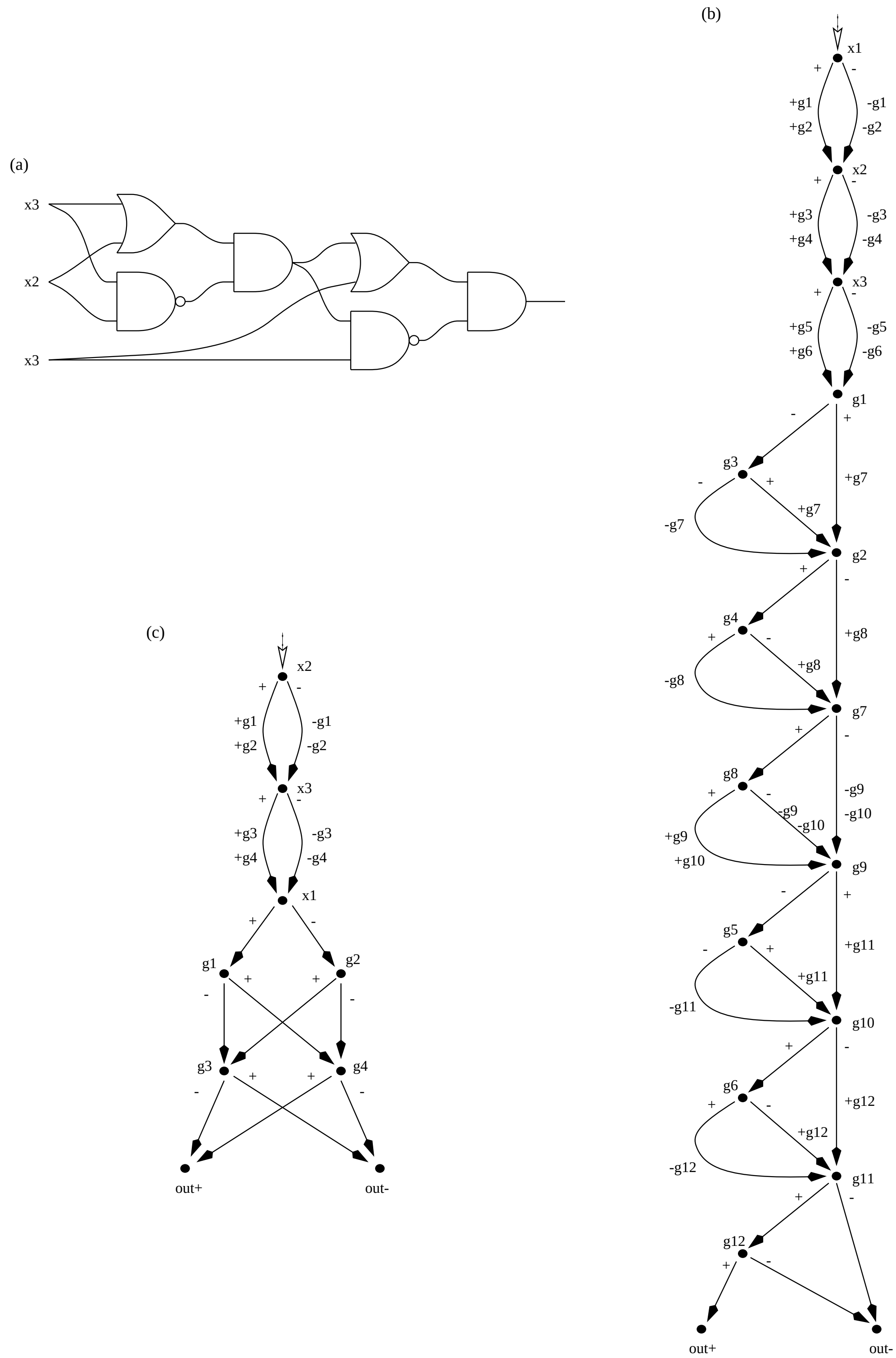

Figure 9: The translation of a 3 input, 6 gate XOR circuit into a μ-WOBP. (a) the circuit, (b) the μ-WOBP generated by our construction, (c) a much simpler μ-WOBP generated by hand.

6 Acknowledgements

I like to thank Takashi Yokomori for inviting me to Chofu, Japan, where this document was conceived; this trip was supported by the Japan Society for the Promotion of Science "Research for the Future" Program, project JSPS-RFTF 96I00101. Also, thanks to Masanori Arita, Daisuke Kiga, Kensaku Sakamoto, Shigeyuki Yokoyama, and Masami Hagiya for discussions; and to Len Adleman for suggesting the name "whiplash PCR" and the HPP example. This work has been supported by the National Institute for Mental Health (Training Grant # 5 T32 MH 19138-07), General Motors' Technology Research Partnerships program, and by the Center for Neuromorphic Systems Engineering as a part of the National Science Foundation Engineering Research Center Program (under grant EEC-9402726).

References

Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

Eric Bach, Anne Condon, Elton Glaser, and Celena Tanguay. *DNA Models and Algorithms for NP-complete Problems*, pages 290–299. IEEE Computer Society Press, 1996.

Charles H. Bennett. The thermodynamics of computation – a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982.

Dan Boneh, Chris Dunworth, Richard J. Lipton, and Jiří Sgall. On the computational power of DNA. *Discrete Applied Mathematics*, 71:79–94, 1996.

R. Deaton, R. C. Murphy, M. Garzon, D. R. Franceschetti, and S. E. Stevens, Jr. Good encodings for DNA-based solutions to combinatorial problems. In Landweber and Lipton (in press).

Masami Hagiya, Masanori Arita, Daisuke Kiga, Kensaku Sakamoto, and Shigeyuki Yokoyama. Towards parallel evaluation and learning of boolean μ-formulas with molecules. In David Wood, editor, *Proceedings of the* 3rd *DIMACS Meeting on DNA Based Computers, held at the University of Pennsylvania, June 23-25, 1997*, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science., Providence, RI, in press. American Mathematical Society.

Laura Landweber and Richard Lipton, editors. *Proceedings of the* 2nd *DIMACS Meeting on DNA Based Computers, held at Princeton University, June 10-12, 1996*, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science., Providence, RI, in press. American Mathematical Society.

Thomas H. Leete, Matthew D. Schwartz, Robert M. Williams, David H. Wood, Jerome S. Salem, and Harvey Rubin. Massively parallel DNA computations: Expansion of symbolic determinants. In Landweber and Lipton (in press).

Qi Ouyang, Peter Kaplan, Shumao Liu, and Albert Libchaber. DNA solution of the maximal clique problem. *Science*, 278:446–449, 1997.

Kensaku Sakamoto, Daisuke Kiga, Ken Momiya, Hidetaka Gouzu, Shigeyuki Yokoyama, Shuji Ikeda, Hiroshi Sugiyama, and Masami Hagiya. State transitions with molecules. In *Proceedings of the* 4$^{\text{th}}$ *DIMACS Meeting on DNA Based Computers, held at the University of Pennsylvania, June 16-19, 1998*, in press .

Willem P. C. Stemmer, Andreas Crameri, Kim D. Ha, Thomas M. Brennan, and Herbert L. Heyneker. Single-step assembly of a gene and entire plasmid from large numbers of oligodeoxyribonucleotides. *Gene*, 164(1):49–53, 1995.

Christopher Y. Switzer, Simon E. Moroney, and Steven A. Benner. Enzymatics recognition of the base-pair between isocytidine and isoguanosine. *Biochemistry*, 32(39):10489–10496, 1993.

Erik Winfree. Complexity of restricted and unrestricted models of molecular computation. In Richard J. Lipton and Eric B. Baum, editors, *DNA Based Computers: Proceedings of a DIMACS Workshop, April 4, 1995, Princeton University*, volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 187–198, Providence, RI, 1996. American Mathematical Society.