

Optimal Bounds on Feasible DNA Computing

Sibabrata Ray
Department of Computer Science
The University of Alabama
Tuscaloosa, AL 35487
sibu@cs.ua.edu

Rajgopal Kannan and S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803
{rkannan,iyengar}@bit.csc.lsu.edu

N. Balakrishnan
Supercomputer Education and Research Center
Indian Institute of Science
Bangalore - 560 012, INDIA
balki@serc.iisc.ernet.in

Abstract: A DNA program may be expressed as a sequence (not necessarily linear) of *ligation*, *extraction* and *combination* steps. As recognized by the researchers in the field, the extraction steps are subject to significant misclassification errors. The error accumulates through the successive extraction steps and may lead to completely wrong conclusions unless measures are taken to control the error.

Many researchers (Lipton, Adleman, Karp, Winfree and others) proposed to limit the misclassification error by repeated application of extraction. All of them assumed that the misclassification probability of a DNA strand is given (or at least bounded) irrespective to the concentration of good strands in the initial tube. In addition, the previous researchers did not consider the probability of false negative and false positive as separate parameters. Further, none of the previous models on error-resilient DNA computing are suitable to study the effect of encoding of the problem on error propagation.

In this paper we are proposing an accurate and analytical model of error propagation in the extraction step that considers the concentration of good strands and probabilities of false positive and false negative explicitly. In addition, the proposed model may potentially be used to study the effect of problem encoding on error resilience.

Introduction and Terminology. DNA is the nature's storage for genetic information. A DNA molecule is a double helix formed by two linear structures called the single strands (or sometimes just strands) of the DNA. Each strand may be viewed as a sequence of sugar molecules connected by covalent bonds in position 3 or position 5. A nucleotide hangs from each of the sugar molecules. There are four types of nucleotides, viz., *Adenine*, *Guanine*, *Thymine* and *Cytosine*, typically denoted by the letters A, G, T and C respectively. Therefore, from the information storage view point, a single strand of DNA is a sequence of letters A, G, T and C. Given a DNA strand, the position 3 of the

last sugar molecule at the one end and the position 5 of the last sugar molecule at the other end are available for forming covalent bonds. This property gives an orientation of the strand called 3'-5' orientation. Adenine molecules can form hydrogen bonds or H-bonds (weaker than covalent bonds) with thymine molecules and guanine molecules can form H-bonds with cytosine molecules. Hence adenine (A) is called complementary of thymine (T) and guanine (G) is called complementary of cytosine (C). This is called *Watson-Crick complementation* or *pairing*. Two H-bonds are formed between A and T and three H-bonds are formed between C and G. Therefore, C-G bonding is much stronger than A-T bonding and plays a vital role in stability of in vitro DNA molecules. If two DNA strands are of same length, oriented in opposite 3'-5' orientation and Watson-Crick complementary in each position, they are called *sticky* DNA. If two sticky DNA come in contact, they form the famous double-helix structure so commonly found in popular science literature. For our purpose though, we'll not attempt to draw the double helix, primarily because it is implied and largely unnecessary, but also for the ease of comprehension and to save labor. Figure 1 illustrates the concepts described here. Note that the half-arrow on individual strands directs toward the 3' end.

Like the 0-1 bits in an electronic computer, A, T, C, G characters in a DNA string are also capable of expressing information. Each test tube of DNA molecules contains huge amount of data (in the order of 10^{14} or so). Adding a chemical to the test tube is equivalent to manipulating all data items using a single instruction, i.e., executing an instruction over a SIMD (*single-instruction-multiple-data*) parallel computer. In an electronic computer we manipulate information in a bit-by-bit (or byte-by-byte) fashion using Boolean logic operations AND, OR, NOT, XOR etc. The information stored in a DNA string may not be separated in convenient pieces very easily and may be manipulated only by chemical reactions, i.e., by adding reagents and by changing environmental parameters like temperature etc. Therefore, the standard SIMD model is not very convenient for understanding DNA computation.

We are following Adleman's model for describing DNA computation. Later other researchers proposed other models for DNA computing [Reif2]. However, the chemistry and lab work involved remain largely unchanged. As per Adleman's description [Adle2] DNA computing begins with a suitable encoding of the entities involved in the problem description. The encoding uses a (improper) subset of the DNA character set $\{A, T, G, C\}$. Once the encoding is done on the paper, commercial laboratories may manufacture DNA molecules representing the encoded data. These artificial DNA molecules built to the specification are called *primers*. The primers are mixed with water, salt, an enzyme called *ligase* and a few other ingredients. If the mixture is kept in proper temperature, the primers in the solution join together to form larger and larger DNA molecules. This process is called *ligation*. It is to be noted that there are two types of ligation, one is called *sticky-end ligation* and the other is called *blunt-end ligation*. Billions of DNA molecules in the solution ligate together forming different larger strands of DNA. If the initial encoding is properly done, some of the new DNA strands (called 'good' strands) are expected to contain the solutions of the problem at hand while others (called 'bad' strands) will not. Ligation of smaller DNA strands is similar to the first part of the computation that takes place in a non-deterministic Turing machine (NDTM). An NDTM computes all possible strings of data that may be generated using the state transition rules and the input data. The second part of the NDTM computation involves choosing the

correct solution out of all the solutions generated. In DNA computation a process called extraction does this. Usually we have a fairly good idea about good strands. Generally we know about the length of the solution, which in turn tells us the length of the DNA strands we are looking for. Further, we'll normally know a subsequence characterization of the good strands, i.e., we'll know a set of subsequences using which only good strands may be isolated. For isolating strands of a particular length a process called *Gel Electrophoresis* is used, where their length using a slab of gel and electrical field sorts the DNA molecules. Extraction using subsequence characterization is more complicated. To extract DNA strands with a particular subsequence a special primer is designed. The primers sequence is complimentary to the subsequence of interest. Iron balls coated with the primer are hung in the solution, a process called *fishing*. DNA strands containing the particular subsequence will form H-bonds with the primer molecules on the iron balls. The iron balls are eventually removed and washed in salt solutions to separate DNA strands.

A few other terms in this context are *restriction enzymes* (cuts a DNA strand at a position defined by a particular subsequence), *annealing* (the process of joining sticky strands to form the double-helix or other double-stranded DNA molecules), *denaturing* (converting double stranded DNA molecules to single stranded DNA), *combining* (mixing contents of several test tubes), *detect* (detecting presence of good strands in a test tube and finding its sequence) etc.

Obviously we lose good DNA strands during extraction due to misclassification as well as due to transfer loss. A mechanism called *Polymerase Chain Reaction* (PCR) is used to generate new good strands in the test tube. PCR works by a series of denaturing, polymerization and annealing. In this paper we do not consider the effect of PCR. We plan to analyze the effect of PCR in our subsequent research.

Problem Description. A typical DNA computation may be described as a *Directed Acyclic Graph* (DAG). Each node of the DAG is a test tube (or tube for brevity). The DAG has exactly one node with in-degree zero. This node (or the source of the DAG) is called the initial tube. The initial tube is the tube containing the mixture of DNA strands right after the ligation step. There may be several tubes with zero out-degree. However, exactly one of them is marked as final or solution tube. The goal of the computation is to obtain a solution tube with a particular volume of DNA strings and a particular concentration of *good* or solution strings. Each tube in the DAG has exactly two out-degrees denoting an extraction step. However, the extraction step need not be simple extraction. It may be a compound or repeated extraction also. A tube with more than one in-degree denotes a combination step. If there is no directed path from a tube to the final tube, then that tube is not useful and may be marked waste tube.

For the purpose of this research we assume that the volume (total number of DNA strings) and concentration (proportion of good strings to the volume) in the initial tube is known. Further, we assume that the volume and concentration requirement of the final tube is also known. We'll show later that the volume and concentration in each tube must maintain a certain relationship. We call a DNA computation feasible, if given the initial volume and concentration; it is possible to achieve the required volume and concentration in the final tube while maintaining the volume-concentration relationship in each intermediate tube.

Generally it is assumed that the good strings may be arbitrarily enhanced by arbitrarily many application of PCR step. However, this assumption is not entirely correct. In addition to the solution strings, PCR may amplify other strings and in addition may generate many unnecessary artifacts in a tube. We plan to analyze the effect of PCR in a subsequent paper. For the purpose of our present work we do not consider PCR step at all.

The earlier researchers modeled the feasibility problem as follows. They assumed that during extraction step each good string moves to the correct tube with a fixed probability p . Hence it is possible to obtain a very high volume very high concentration 'yes' tube by repeating the simple extraction step a few times. They assumed that the extraction step is standard and performs identically independent of volume, concentration and other parameters.

Related Works and Their Limitations. From the preceding discussion it is clear that a DNA computation algorithm may be described as a sequence of extract, combine and detect primitives. PCR steps may be introduced in any place of the algorithm and is believed that will only be beneficial by increasing the number of good strands. Winfree [Winfree1] called the PCR step as *duplicate* step. He formalized compound extraction step by representing it as a single-source two-sink directed acyclic graph (DAG). (It is to be noted that earlier we modeled the whole computation by means of a DAG.) Here we present a slightly enhanced form of Winfree's model. The source of the DAG is the test tube containing good and bad strands whereas the sink nodes are two test tubes, one containing good strands and the other containing bad strands. All other nodes represent the extract or duplicate operation. If a node has two incoming edges there is an implied combine operation in that node. For each extract node there are two outgoing edges labeled σ and σ' . $S(\sigma)$ is the subsequence that is used to extract DNA at this node. The edge labeled σ refers to the test tube with DNA strands including $S(\sigma)$ and the edge labeled σ' refers to the test tube not containing $S(\sigma)$. Duplicate nodes have only one outgoing edge.

Clearly, each extraction step introduces both the false positive and false negative errors. In other words, some of the good strands will be classified as bad strands (type I error) and some of the bad strands will be classified as good strands (type II error). The errors will be accumulated through a large number of steps and finally may lead to wrong result. While it is important to invent new extraction technologies to reduce errors in strand classification, computer scientists have made several attempts to make the existing methods more error resilient by combinatorial methods for repeated extraction and better problem design.

The computing community primarily concentrated on variations of three methods of error reduction. Two of those were proposed by Leonard Adleman [Adle2] and the third one was proposed by Lipton and others [Lipton2]. Digestion based techniques are primarily applicable to reduced volume computation. As we are interested in general DNA computing paradigm, we'll not discuss digestion-based methods here. Adleman's first method involves repeated application of the extraction step on the 'yes' tube, i.e., the tube holding DNA strands containing the subsequence $S(\sigma)$. If the false negative rate is high (i.e., probability of misclassification of a good strand as a bad strand is significant)

this method will result in significant loss of good strands. Karp and others [Karp1] proposed a method to improve Adleman's method of repeated extraction. He proposed to simulate reliable extraction method by multiple application of existing faulty extraction methods. This is called compound extract. Karp claimed that if δ is the desired error probability of the extract method and ϵ is the error probability of the available method, then error rate δ might be achieved by $O(\log_{\epsilon}^2 \delta)$ applications of the available method in $O(\log_{\epsilon} \delta)$ parallel steps. Chen and Winfree [Winfree1] formalized Karp's method and proved that the constants involved are not large. We discuss Winfree's method here, as that is the culminating point of this approach.

Figure 2 explains Winfree's method by providing an example. Suppose the DNA strands are to be extracted using the subsequence s . The source node of the DAG represents the initial test tube. The extraction in the first level provides two tubes. The extraction method is applied on each tube and thus three tubes are obtained, the middle tube is a combination of one yes and one no tube coming from left and from right respectively. The extraction steps may be repeated arbitrarily many times. It is to be noted that Winfree assumes that the error rates of the extraction method remain largely same from one layer to another layer and between test tubes in a single layer, an assumption we wish to contradict later.

The second approach proposed by Adleman involves using PCR to amplify the number of good strands in a test tube. If the good strands are lost between steps then this mechanism may be used to replenish the supply of good strands. Boneh and others [Lipton2] proposed and analyzed the use of PCR to reduce the error rate in case of decreasing volume computation (where the bad strands are discarded as soon as they are discovered). However, many important problems like formula-SAT and breaking DES are not decreasing volume computation. Again Chen and Winfree extended Boneh's method to the general bio-molecular computation paradigm. They assumed that all the strands are to be retained and may be used in subsequent steps. Using the assumption that PCR will amplify the number of good strands without significantly amplifying the number of bad strands they gave methods to compute the frequency of PCR applications for a desired success probability. It is to be noted that the performance of PCR will depend on the structure of bad strands and their concentration. Hence Chen and Winfree's method needs to be modified to give better bounds. In fact in some cases the application of PCR may not be only beneficial and hence may not be advisable at a high frequency.

The third approach (proposed by Lipton and others [Lipton2]) attempts to improve error resilience by choosing good encoding of data. In particular they propose to use encoding such that each strand will contain either $S(\sigma)$ or $S(\sigma')$ but not both. This approach is good but the results were not developed sufficiently. Among other things, this approach increases the size of each strand and therefore limits the size of the problems those may be handled by using smaller DNA strings. Further, if the concentration of bad strands is high and there is a long common subsequence between $S(\sigma)$ and some other part of the bad strands (or vice versa) then this simple coding method may not be very helpful. Attempting to avoid long common subsequences will increase the problem/solution size even further and should not be attempted without a clear idea of trade-offs.

Analysis of Simple Extraction Step. A close look at the extraction by fishing will convince one that the assumption of constant (or bounded) misclassification probability of good DNA strand is not a good assumption. To fish good strands from a tube, one needs to dip iron balls coated with suitable primer molecules in the tube. The primer molecules, if suitably designed, will anneal with the good strands and come out with the iron balls. If only good strands could anneal with the primer molecules then there would be no problem and the simple extraction mechanism would have worked perfectly all right. However, annealing is a blind mechanism. A primer molecule will bind with any strand with a properly oriented and long complimentary subsequence. Therefore, in practice, the bad strands may also anneal with primer molecules. But the probability of annealing with a bad strand is supposed to be lower in comparison to the probability of annealing with a good strand.

Therefore, fishing a molecule from a test tube involves two steps. In first step, the primer molecule needs to meet the intended molecule and in the second step the primer needs to anneal with the intended molecule. It is to be noted that meeting a molecule with an appropriate complimentary subsequence does not mean that the annealing will always occur. Meeting of the primer and the molecule does not guarantee meeting of the complimentary sections of the molecule and the primer. Therefore, we model the situation by assigning a probability of annealing given that the primer and the molecule met. In particular, we model the situation using two different probabilities. The first is the probability of annealing to a good molecule after meeting, denoted by p , and the second is the probability of annealing to a bad molecule after meeting, denoted by p' . Typically the value of p will be 0.9 or greater and the value of p' will be 0.2 or smaller. The actual value of p and p' will depend on several factors like the structures of good strands, bad strands and primers, temperature and other physical and chemical parameters etc. A good molecule may not anneal to a primer molecule after meeting with probability $q=1-p$. Further, we define $q'=1-p'$. When a primer anneals to a bad strand, we call it the error of false positive or type I error. When a primer fails to anneal to a good strand, we call it the error of false negative or type II error. In our model, p' is the probability of type I error and q is the probability of type II error.

A simple extraction step begins with an initial test tube containing a mixture of good and bad strands. The good strands are characterized by some subsequence present in all good strands and absent in all bad strands. Good strands are fished out from the initial tube and stored in a tube called 'yes' tube. Once the fishing is over, the remaining tube is supposed to contain mostly bad strands and called 'no' tube. Let V_I denote the total number of strands in the initial tube and R_I denote the ratio of the number of good strands to V_I in the initial tube. V_I is expressed in moles and R_I , being a ratio of number of molecules, is unit free. Further, V_Y , R_Y , V_N and R_N are the total number of strands and ratio of good strands in yes and no tubes respectively.

At this point of time we consider a thought experiment. Let us try to fish exactly one more molecule from the no tube to the yes tube. We dip exactly one molecule of primer in the no tube and let it be in the tube until it anneals with a molecule. Whatever strand it anneals with is transferred to the yes tube. Let P_G denote the probability that we'll indeed transfer a good strand to the yes tube. The primer may meet a good molecule in its first chance and anneal with it. In that case a good strand will be transferred to the yes tube. Otherwise, the primer may not anneal to anything in its first meeting, but then it meets a

good molecule in its second meeting and anneal to it. Then also a good strand is transferred to the yes tube. In general, to transfer a good strand to the yes tube, the primer may not anneal to anything from $1, \dots, i-1$ meeting and then anneal to a good molecule in the i^{th} meeting for $i=1, \dots, \infty$. The probability of meeting a good molecule is R_N assuming that the good strands are randomly distributed in the tube. The probability of not annealing with any strand in a particular meeting is $R_N q + (1-R_N)q'$. Therefore.

$$P_G = R_N p + (R_N q + (1-R_N)q')R_N p + (R_N q + (1-R_N)q')^2 R_N p + \dots = \frac{R_N p}{p' - R_N(p' - p)}.$$

At this point of time we may entertain three possibilities, $P_G > R_Y$, $P_G < R_Y$ and $P_G = R_Y$. Further, the goal of the extraction step is to increase R_Y and decrease R_N . Therefore, if $P_G > R_Y$, transferring the newly fished strand to the yes tube will increase R_Y and should be done. Similarly, if $P_G < R_Y$, then the newly fished strand should not be transferred to the yes tube. Not only that, if there are a very large number of strands in the solution (i.e., if we assume that the parameters are continuous) then $P_G < R_Y$ implies that we should not have fished the previous strand also. Therefore, $P_G = R_Y$ gives us a local optima and any well designed simple extraction step must obey the equation. By substituting the expression of P_G in the equation we get

$$R_N p = R_Y (p' - R_N(p' - p)).$$

In addition, note that the total number of good strands in the initial tube is same as the sum of good strands in the yes tube and no tube. The same relation holds for the bad strands also. By writing the expressions for the preservation of strands and by algebraic manipulation we get

$$\frac{V_Y}{V_I} = \frac{R_I - R_N}{R_Y - R_N}.$$

It is to be noted that in the previous equations we ignored the transfer loss and loss due to finite amount of time allowed for fishing. We plan to address these two types of losses in our future work.

Two more inequalities govern a simple extraction step. We may reasonably expect that the no tube will have lower concentration of good strands in comparison to the initial tube. Further, more and more good strands are fished out, P_G reduces further and further. Therefore, the value of P_G is maximum at the beginning of the extraction step. By putting all these equalities and inequalities together we get the following theorem.

Theorem 1. A well designed simple extraction step will be governed by the following relations.

1. $R_N p = R_Y (p' - R_N(p' - p))$
2. $\frac{V_Y}{V_I} = \frac{R_I - R_N}{R_Y - R_N}$
3. $R_N \leq R_I$ and
4. $R_Y \leq R_I p / (p' - R_I(p' - p))$

Figure 3 illustrates theorem 1, where $p=0.90$, $p'=0.15$, $R_I=0.5$ and $V_I=1.0$. The curves in the figure show the values of R_Y and V_Y for values of R_N in the feasible range. As one can see from the figure, and theorem 1, given p , p' , R_I and V_I , one may choose exactly one of the parameters R_Y , V_Y , R_N , V_N . The other parameters are automatically decided by the relations in the theorem 1.

Feasibility of DNA computing algorithms. Theorem 1 may be used to decide whether a particular DNA computation is feasible. We demonstrate our idea by using an example. For the purpose of our example we assume that the extraction steps are simple extraction steps. But we'll see in the next section that the same mechanism may be used to decide the feasibility of DNA algorithms involving compound extraction steps.

Consider the DNA computation described in figure 4. It involves nine tubes, T_1, \dots, T_9 . T_1 , T_3 , T_4 and T_7 are source tubes in an extraction step. T_1 is extracted using the subsequence $x1$ and T_2 is the yes tube and T_3 is the no tube. We know the probabilities of annealing to a good strand and bad strand at this step ($p1$, $p1'$). The corresponding probabilities at T_3 , T_4 and T_7 are $p3$, $p3'$, $p4$, $p4'$, $p7$, $p7'$. The DAG represents the DNA computation for the SAT problem ($x1$ or not $x2$) and ($x2$ or not $x3$). As we discussed in the previous section, we may choose the ratio of good strands in yes tube at each of the intermediate step. For example, R_2^{up} (R_2^{dn}) is the ratio of the strands containing $x1$ ($x2$) in T_2 and V_2 is the total number of strands in T_2 .

We assume that we started with a tube containing random strands of length three or less (as there are only three literals). Therefore, the distribution of strands in T_1 is known. In particular, we know the probabilities like $p(x1)$ (proportion of strings containing $x1$) or $p(x1 \ x2')$ (proportion of strings containing $x1$ but not $x2$). Further, let us assume that we have an upper bound known for the strands encoding a solution. In other words, suppose ps be the proportion of the strands encoding a solution to the problem.

Here we estimate the probability that a solution strand will be discarded improperly without reaching the tube T_4 (i.e., by reaching T_5) may be estimated by the

formula
$$\frac{R_3^{up} \times R_5^{up}}{R_1^{dn} \times R_3^{dn}}.$$

Similarly the probabilities that a solution strand will be discarded by reaching T_9 given that it reached T_4 may be computed. The probability that a non-solution strand will reach T_8 may be estimated using similar technique.

We call the computation feasible if the final tube (here T_8) contains a certain volume of strands and a particular proportion of solution strands. At this point of time the problem reduces to a numerical search problem where we search the feasible space for R_i^{up} and R_i^{dn} to find that whether we can exceed the threshold of feasibility.

Analysis of Compound Extraction. So far we have discussed only the simple extraction. It is possible to improve the performance of extraction step by repeated application of simple extraction. A compound extraction is described by the DAG model given by Chen and Winfree. We do not have a complete analysis of a general compound extraction step available yet. The following example will illustrate our idea for analyzing compound extraction using our method.

Consider the two-step compound extraction step in figure 5. T_1 is the initial tube, T_4 and T_6 are yes and no tubes respectively. V_i and R_i are volume of all strands and concentration of good strands in T_i respectively, $i=1, \dots, 6$.

Note that V_1 and R_1 are given. Once we choose V_3 and V_5 , we can compute the V_i , R_i for all tubes using the equations in theorem 1. This gives us a parametric relationship between V_4 , R_4 , V_6 , R_6 . This parametric relation governs the particular two-step compound extraction.

The method described here may be generalized for bigger compound extraction steps also. We are currently working in that direction.

Incorporating effect of encoding in error analysis. The probabilities p and p' in this paper denotes the probability of annealing of a good/bad strand with the primer upon meeting. If the physical and chemical environment is given, then p and p' are functions of the sequences of good/bad strands and primer. Currently we are working to find functional forms of p and p' . This research will help us to judge the merit of initial encoding schemes.

Conclusion. When fishing good strands from a tube, the relative concentration of good strands will affect the chance of fishing. In addition, the initial encoding of data will affect the probabilities of type I and type II errors. To the best of our knowledge this research is the first effort to formally consider the effect of concentration and type I and type II errors in simple extraction and gives the governing equations for a well-designed simple extraction step. We are currently working on applying our method in compound extraction and feasibility study of DNA algorithms. The results reported are works in progress.

References.

- [Adle1] Leonard M. Adleman, "Molecular Computation of Solutions to Combinatorial Problems", *Science*, 266:1021-1024, Nov. 11, 1994.
- [Adle2] Leonard M. Adleman, "Computing with DNA", *Scientific American*, 34-41, Aug. 1998.
- [Adle3] Leonard M. Adleman et. Al., "On Applying Molecular Computation to the Data Encryption Standard"
- [Gehani1] A. Gehani, T. LaBean and J. Reif, "DNA-Based Cryptography", <http://www.cs.duke.edu/~reif/paper/DNAcrypt/crypt.pdf>.
- [Lipton1] Richard J. Lipton, "DNA Solutions of Hard Computational Problems", *Science*, 268:542-544, 1995.
- [Lipton2] Dan Boneh et. Al. "Making DNA Computers Error Resistant", *DNA Based Computers II, DIMACS Workshop*, 1996, 163-170.
- [Winfree1] Kevin Chen and Erik Winfree, "Error Correction in DNA Computing: Misclassification and Strand Loss", http://www.dna.caltech.edu/~winfree/old_html/papers/kevin.ps.
- [Reif1] J. Reif and T. LaBean, "Computationally Inspired Biotechnologies: Improved DNA Synthesis and Associative Search Using Error-Correcting Codes and Vector-Quantization", <http://www.cs.duke.edu/~reif/paper/Error-Restore/Error-Restore.ps>.

[Reif2] J. Reif, "Paradigms for Biomolecular Computation",
<http://www.cs.duke.edu/~reif/paper/paradigm.ps>.

[Karp1] R. Karp et. Al., "Error-Resilient DNA-Computation", *Proc. ACM-SIAM Symposium on Discrete Algorithms*, Jan 28-30, 1996, Providence, RI, 458-467.

Illustrations.



Figure 1: Two sticky strands of DNA annealed together.

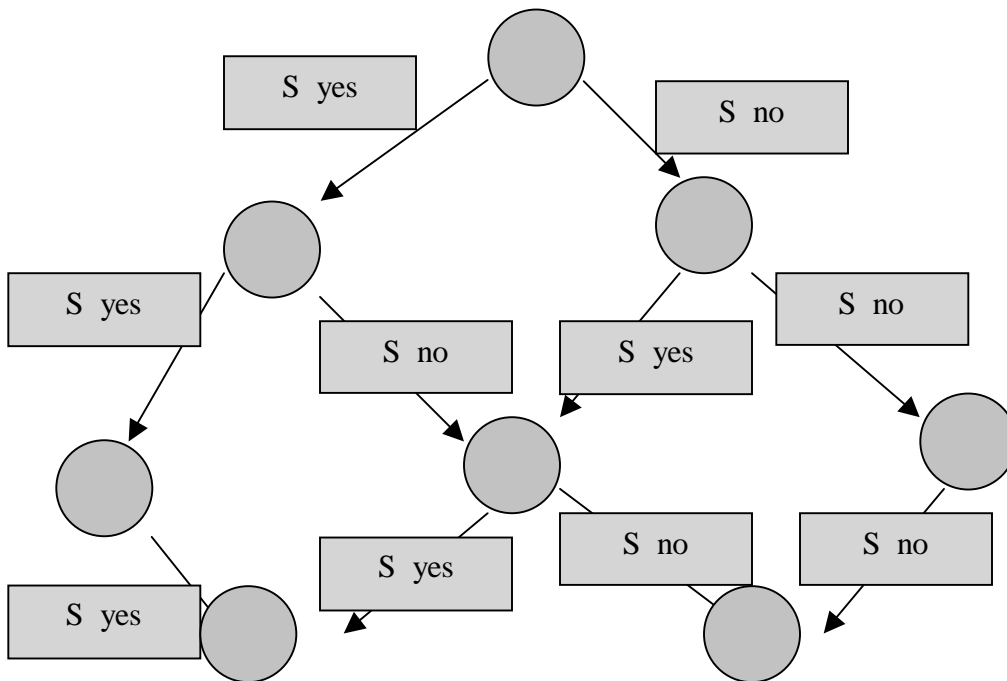


Figure 2: Step Diagram for a 2-step compound extraction

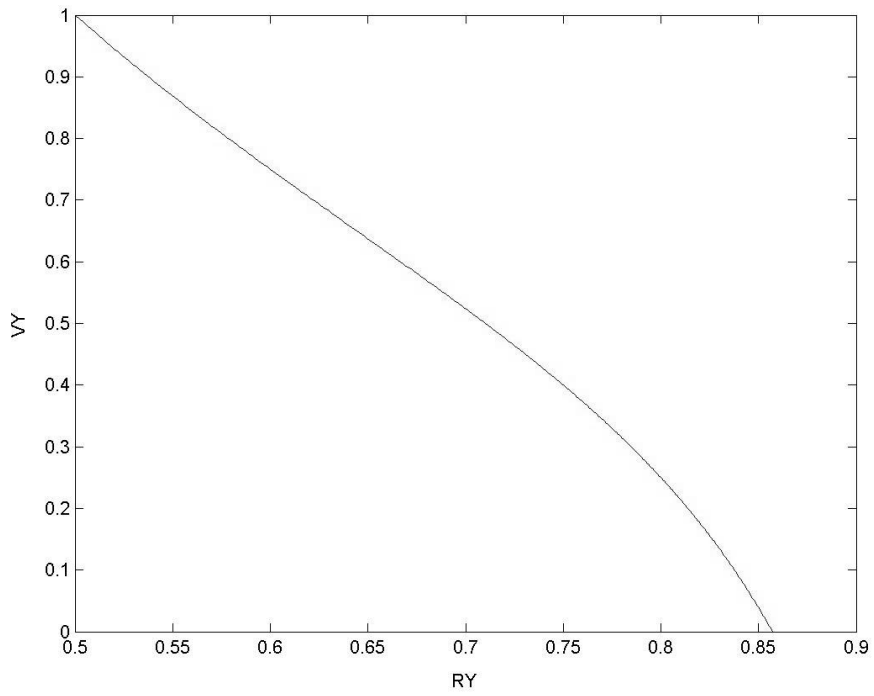
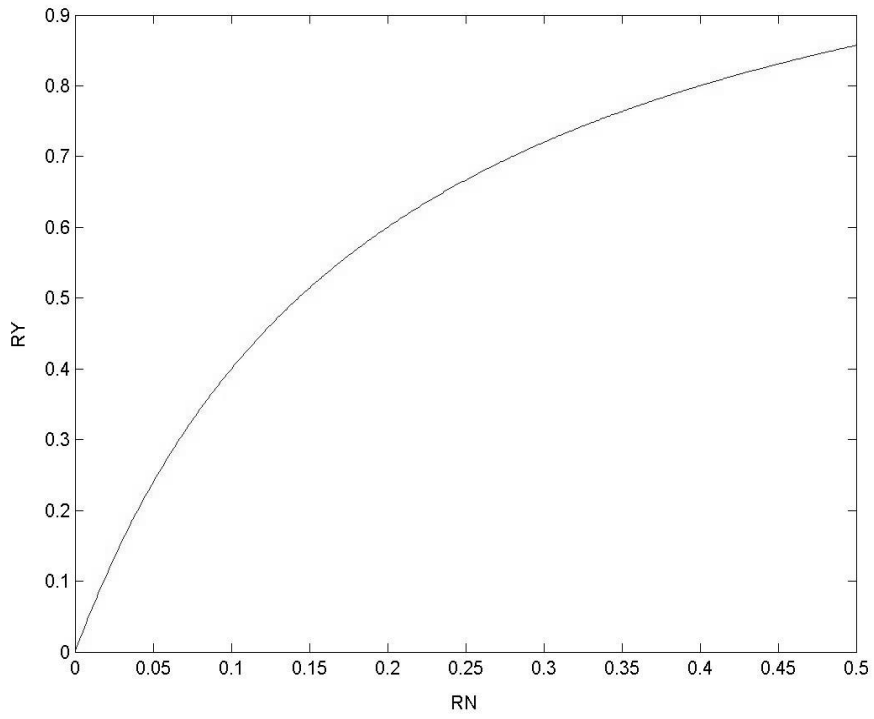


Figure 3a and 3b: Theoretical relationship between parameters of a simple extraction step

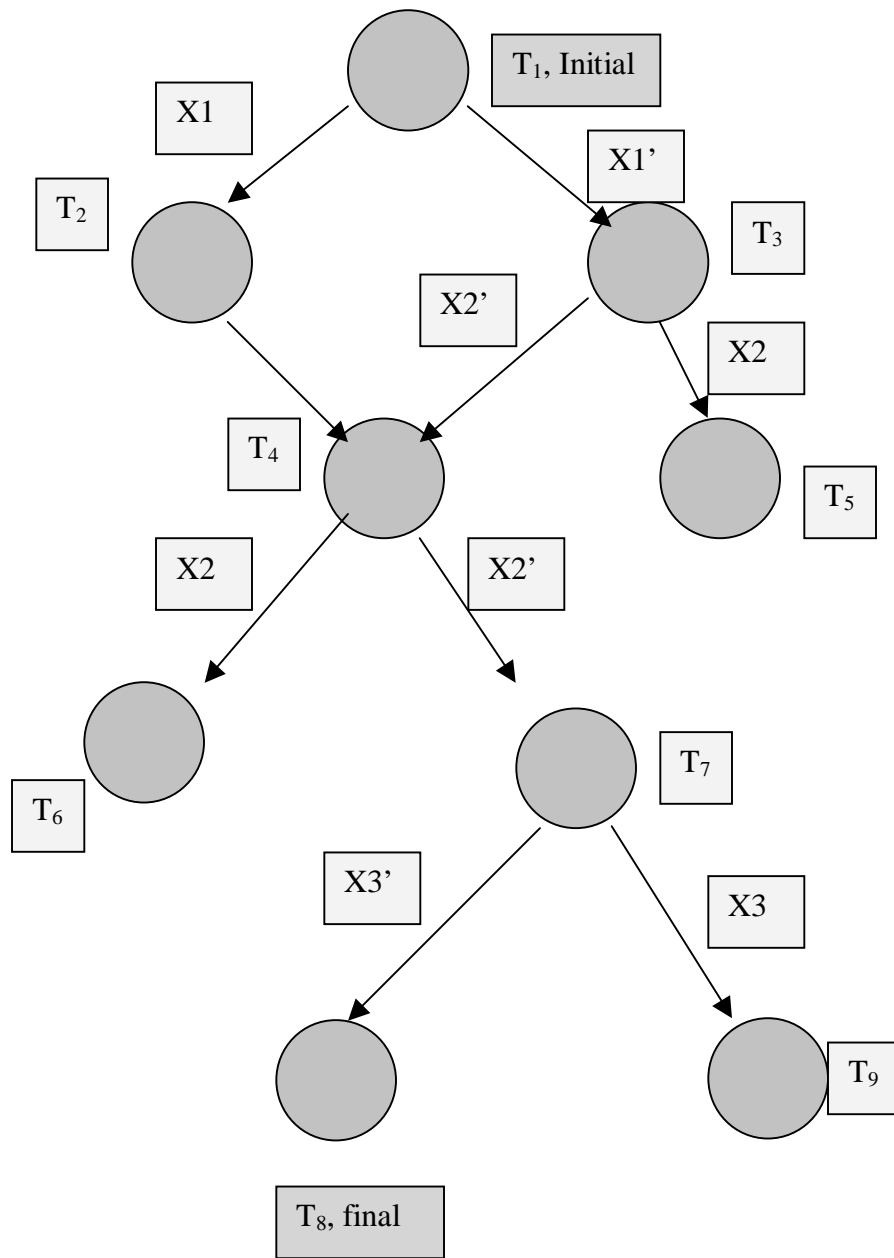


Figure 4: DAG representation of a DNA algorithm

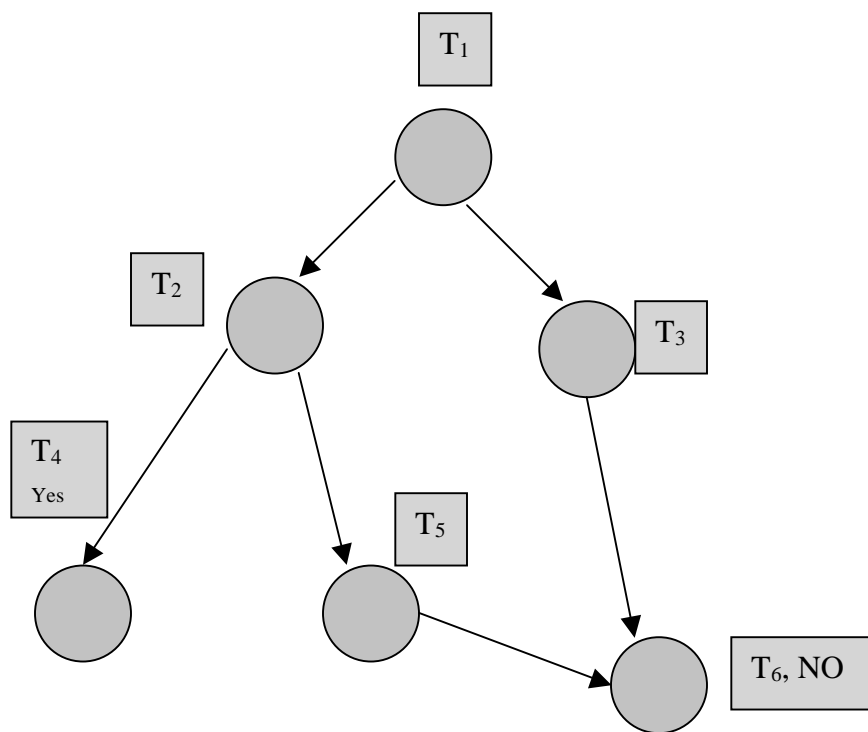


Figure 5: A two level compound extraction step