Simulating Biological Reactions: A Modular Approach

Alexander J. Hartemink, Tarjei S. Mikkelsen, and David K. Gifford

Abstract. We develop a general framework for simulating a sequence of biological reactions using small simulation modules. We demonstrate the usefulness of such a framework by implementing a simulator called the cybercycler. The cybercycler contains DNA hybridization, polymerization, ligation, and melting modules linked together to simulate a thermocycling process. This simulator enables us to interpret the behavior of our own programmed mutagenic unary counter in the laboratory. We describe the modules we implemented and then present a comparison of output from the cybercycler with the results of laboratory experiments to evaluate the effectiveness of the cybercycler simulator. We define a concrete specification for transformation modules and for populations of molecules passed between modules. This specification enables the construction of tools for simulating an arbitrarily complex sequence of biological reactions. We provide Java interfaces and classes for building such tools.

1. Introduction

Biological reactions implementing any non-trivial form of DNA-based computation are likely to be extremely complex: computations intended to solve large problems will require large populations of molecules, and the biological reactions used to carry out the computations will transform these populations in complicated ways. The prospect of reasoning about all the products being formed in these reactions is quite daunting.

In our laboratory, we have constructed a simple unary counter to experimentally demonstrate the feasibility of DNA-based computation using programmed mutagenesis [**12, 14**]. While the counter has been engineered to operate cleanly [**13**], the repeated thermocycling that increments the counter causes a wealth of product species to be formed after just a few cycles. This reaction complexity makes it difficult to predict or even interpret the results of frequent thermocycling; even when the results are predictable and interpretable, simply keeping track of the myriad interactions taking place during the reaction remains a significant challenge.

1991 *Mathematics Subject Classification.* Primary ; Secondary .

Hartemink is partially funded by a National Science Foundation Graduate Research Fellowship.

Tools that can predict, analyze, interpret, and keep track of the products being formed in biological reactions would be of great benefit. In that context, the contribution of this work is three-fold: first, we develop a general framework for building tools to simulate biological reactions; second, we present the CYBERCYCLER, a tool built within this framework and able to simulate the reactions necessary for understanding our laboratory unary counter implementation; and third, we provide Java interfaces and classes that define a concrete specification for this framework, enabling the construction of tools for simulating an arbitrarily complex sequence of biological reactions. The CYBERCYCLER demonstrates that this general framework is useful for building simulation tools, and the Java interfaces and classes instantiate this framework, producing a flexible and extensible environment for building simulation tools more easily.

Overview of this Paper. We begin by surveying some related work to motivate our research and then develop a general framework for building tools to simulate biological reactions. Once the basic elements of this framework have been elucidated, we present the CYBERCYCLER, a tool for simulating thermocycled reactions such as those incrementing our unary counter [**12, 14, 13**]. We briefly outline the various modules we implemented for the CYBERCYCLER, and then compare the output of the CYBERCYCLER with the results of laboratory experiments to evaluate the effectiveness of the CYBERCYCLER simulator. Finally, we offer a collection of Java interfaces and classes as a specification for building general simulation tools. We close by suggesting some directions for future work.

2. Related Work

The need for rational design and analysis of biological reactions has spawned a variety of software tools. These tools can generally be divided into three categories: thermodynamic prediction of secondary structure, thermodynamic analysis of hybridization (including optimal PCR primer selection), and simulation of specific models of DNA-based computation.

The *mfold* algorithm developed by Zuker, *et al.* [**31**] is perhaps the most notable program using thermodynamics to predict secondary structure of single-stranded DNA and RNA.

Tools in the second category include BIND by Hartemink, *et al.* [**12**], Oligo from MBI, Inc. [**15**], HYBsimulator from AGCT, Inc. [**2**], and the recent HYTHER program developed at the SantaLucia laboratory [**18**]. PCRsim by Rubin, *et al.* [**22**] and the Amplify program by Engels [**9**] simulate standard PCR reactions.

Among the various tools developed for the purpose of simulating specific models of DNA-based computation, Reif, *et al.* [**21**] have developed a graph-theoretic model for simulating recombinant DNA reactions, and Hagiya, *et al.* [**10**] have developed an applet for analyzing their DNA computing algorithms. Hartemink, *et al.* [**13**] wrote SCAN to optimally select nucleotide sequences according to given design constraints.

We are unaware of previous work describing the simulation of arbitrary sequences of DNA hybridization, polymerization, and ligation reactions. Though each of these reactions has been studied and simulated separately, we were unable to combine available tools so as to use them in concert.

3. General Framework for Simulation

While a plethora of software tools exist to simulate specific reactions, there is a profound lack of tools designed to simulate arbitrary sequences of biological reactions. A framework that effectively facilitates such simulation must be flexible, extensible, and general. By developing an "open source" framework with these properties, we hope to do more than construct a simulation tool; we hope to offer a means for easily building and expanding simulation tools.

3.1. Modular Design. In our framework, we represent a complicated sequence of reactions as a sequence of transformations on molecular populations. Populations are the material of a reaction, while transformations are the agents of change in a reaction. Transformations can represent active agents (*e.g.*, enzymes), environmental changes (*e.g.*, temperature increase), mechanical operations (*e.g.*, sampling or splitting populations), or filters (*e.g.*, micropore).

Transformation modules are implemented according to a concrete specification which allows them to be composed with one another. As a result, complex reactions can be simulated by building appropriate transformation modules and then linking them together in a sensible way.

This framework provides a tool-building environment which is flexible and completely general. It encourages code reuse, but remains fully extensible: researchers can tailor or tweak modules according to their needs without having to rewrite entire simulators. As new data or models for various transformations become available, modules can be updated and redistributed easily.

3.2. Taggable Data. We permit every species within a population to be associated with an arbitrary number of tags. Tags on species can be used to facilitate reaction simulation (*e.g.*, a tag indicating that a particular species is immobilized, allowing it to be separated from other species that are not), experimental analysis (*e.g.*, the history tag, discussed below), or output functionality (*e.g.*, the radio-label tag, also discussed below). Transformations may read or write tags in performing their respective duties, and are responsible for updating them as necessary.

4. CYBERCYCLER Design and Implementation

The CYBERCYCLER accomplishes two purposes. Primarily, it helps us better understand our unary counter by predicting the products being formed during the reaction as it is thermocycled. Secondarily, it serves as a "proof of concept" for our general simulation framework.

To build the CYBERCYCLER within our general simulation framework, we make a few simplifying assumptions. We assume that polymerase and ligase can be modeled as though they are acting serially even though they are both present simultaneously in the reaction. In the case of polymerase and ligase, this seems reasonable. Similarly, we assume that hybridization and polymerization can be modeled in a serial fashion. This seems justified by the fact that PCR frequently consists of a low temperature "annealing" step followed by a raised temperature "extension" step for polymerization, segregating the two operations so as to minimize non-specific binding.

In the context of our general framework, we implemented transformation modules representing melting, hybridization, polymerization, ligation, and CyberGel output. As shown in Figure 1, the first four of these modules are linked together

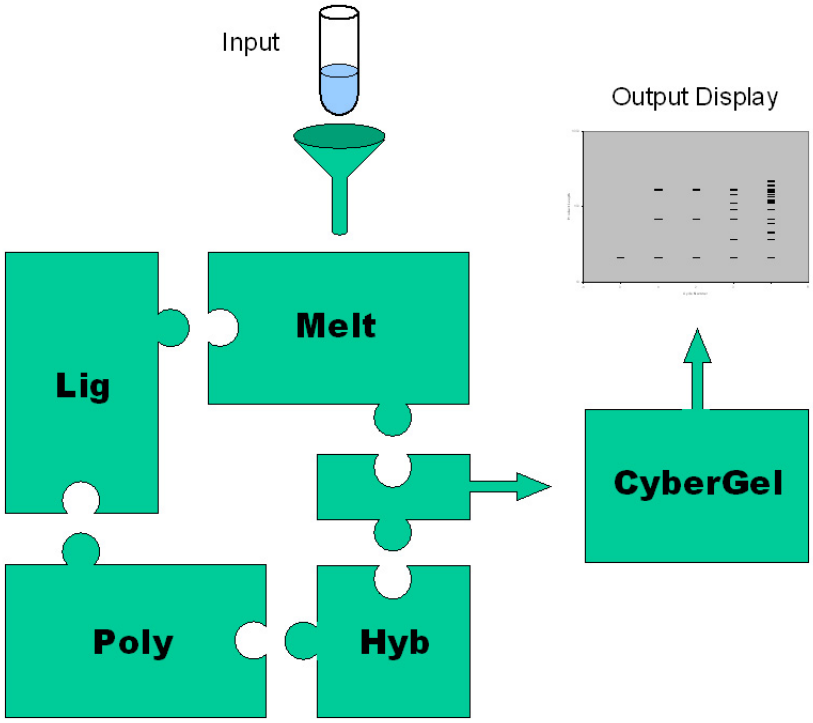

FIGURE 1. *A schematic look at the* CYBERCYCLER *simulator: In this depiction, various modules are composed to form the* CYBERCYCLER. *The initial population enters the melting module as input, and the resulting population is passed in turn to the hybridization module. The polymerization module receives its input from the hybridization module and passes its result to the ligation module, which cycles its result back to the melting module. The population is sampled after each time through the melting module, with the sample going to the CyberGel module for display.*

in a loop, and the population products are sampled and passed to the CyberGel module after each cycle.

Furthermore, we tag each strand in the population with a radio-label tag and a history tag. The radio-label tag is a simple Boolean tag indicating if a strand has been labeled with radioactive ^{32}P. Any labeled strand that is polymerized or ligated yields a labeled strand. In the CyberGel module, only labeled strands produce bands, enabling the output to be compared with autoradiographs from laboratory experiments.

The history tag describes the reaction in which a strand was formed. Strands present in the initial population have empty history fields, but any strand formed during a reaction is tagged with a string describing how it was formed. This allows every strand's pedigree to be tracked throughout the simulation, a form of analysis that is impossible in a laboratory setting.

In the following subsections, we describe the various CYBERCYCLER modules and the modeling choices we made in their implementation.

4.1. The Hybridization Module. The hybridization module takes a population as input and numbers each single-stranded species of DNA from 1 to n, where n is the total number of single-stranded species of DNA in the population. Then for each $i \in \{1, \ldots, n\}$ and each $j \in \{i, \ldots, n\}$, the module considers the possible hybridization of strand i with strand j at every offset position between the two strands[1]. For each of these strand×strand×position possibilities, the module calculates the predicted ΔH° and ΔS° for the hybridization reaction.

Predictions of enthalpy and entropy are calculated using the nearest neighbor stacking model [**12, 23, 24, 26, 20, 8**]. Thermodynamic parameters used in the calculation of ΔH° and ΔS° are the most recent published, taken from [**23, 17, 3, 4, 5, 6**]. Hybrid melting temperature is computed from ΔH°, ΔS°, strand concentrations, and cation concentration, as described in [**12, 29**]. The hybridization module is parameterized over the annealing temperature, and it considers as valid any binding whose melting temperature exceeds the annealing temperature.

For our purposes, a conservative (overstated) estimate of the possible products in the unary counter reaction is desirable. Therefore, for the sake of computational efficiency, our module implementation does not employ a chemical kinetic model to determine product concentrations, but instead uses only the simple heuristic that any new strand is formed at a concentration equal to the lowest concentration of any of the reactants involved in its formation. This tends to overstate strand concentrations, but the effect of overstated concentrations on predicted melting temperature is almost negligible and errs in the direction of conservative product estimation.

Different applications may require simulations that compute more accurate values of product concentrations, but this is easily achieved as a result of our modular framework: a re-implemented hybridization module could incorporate reaction dynamics to determine how much of each reactant will be used in forming each possible product[2].

4.2. The Polymerization Module. The polymerization module checks each strand in the population for bound strands present after the hybridization step. For each of the m bound strands, the module checks for a stable $3'$ end and if present, extends the bound strand until it reaches the end of the template, or another bound strand blocking its path. As before, our implementation is conservative in that it predicts extension products representing polymerization to each of these possible blockage lengths.

We assume that the polymerase exhibits perfect fidelity (it introduces no mismatched nucleotides, insertions, or deletions), full extension of the strand product

[1] This includes the cases where one strand extends past the end of the other strand; in total, the module considers length(i) + length(j) - 2 possible hybridization positions. Note also that the module considers the case where j is equal to i in order to check for possible dimer formation. Though the module does not currently check for monomeric secondary structures such as hairpins, our population specification is perfectly capable of representing such structures.

[2] See [**11, 25, 28**] for a more detailed discussion of reaction dynamics and chemical kinetics. In general, accurate prediction of product concentrations may be quite complicated since there are competitive reactions and multi-strand hybridization possibilities. For a modestly-sized population, however, it should not be too difficult to achieve a reasonable approximation of product concentrations.

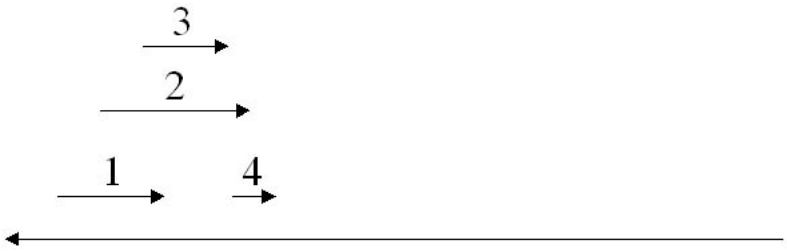

FIGURE 2. *Four binding strands may overlap: In this depiction, some of the binding strands overlap with one another in such a way that certain strand combinations preclude simultaneous binding. A naïve implementation might consider all 15 non-empty strand combinations, but our recursive implementation considers only the 6 that do not result in strand overlap.*

(until it encounters either a blocking strand or the end of the template), and no exonuclease activity (either $3' \rightarrow 5'$ or $5' \rightarrow 3'$).

In generating extension products corresponding to all possible blockage lengths, our module implementation incorporates a recursive polymerization procedure. Starting from the strand located most proximate to the $3'$ end of the template, the procedure calls itself again twice, once incorporating the strand, and once without it. In the recursive call where the strand is incorporated, the next strand considered is the first one whose $5'$ end is beyond the $3'$ end of the current strand. In the recursive call without the strand, the next strand considered is the next one in the $5'$-sorted list. This recursive procedure allows large portions of the binary subset tree to be pruned.

In the case shown in Figure 2 for example, the number of considered combinations is reduced from $2^m - 1 = 15$ to 6. As m grows and the degree of overlap increases[3], the computational savings becomes even more significant. Anecdotally, running times for simulations of five-cycle reactions with this recursive procedure were around a minute, whereas running times for an iterative version of the module were around an hour.

4.3. The Ligation Module. The ligation module receives the intermediate population[4] from the polymerization module and looks for possible ligation sites:

[3]Overlap becomes increasingly common in later cycles as polymerase generates new species from subsequences of existing species in every cycle.

[4]The population of strands that is passed between the two modules has no real significance and exists only because of our modeling assumption that we can treat the two as separable modules and apply them serially. We refer to such a population as an *intermediate pseudo-population*, since it is purely an artifact of the modeling. Our claim in modeling the reaction this way is that the population that results from applying the polymerization and ligation modules in series is a reasonably accurate approximation of the one produced if the modules were to be applied simultaneously, but of course, an artifactual pseudo-population may be produced along the way. This pseudo-population should not be considered as existing in a real experimental setting.

adjoining strands bound to a single template. If the $3'$ end of the upstream strand and the $5'$ end of the adjacent downstream strand are both stable, the module ligates the pair together to produce a longer strand.

Once again, the ligation module is conservative in that it assumes that any putative ligation can either fail or succeed and reports both possibilities as products. For our purposes, it suffices to assume that half the putative ligations fail and half succeed, so as to allow the CYBERCYCLER to consider both possibilities. Applications needing a more accurate measure of ligation efficiency are easily accommodated by a simple re-implementation of the module.

4.4. The Melting Module. Like the hybridization module, this module is parameterized over temperature. However, when the temperature is higher than the highest temperature permissive for nucleic acid hybridization, the actual implementation of this module is trivial. When that assumption holds, as is the case in our unary counter, the module simply examines the various species in the population and, for each one that is double-stranded, it produces two single-stranded species. The resulting population is then passed to the next module in the chain. In the case of the CYBERCYCLER, the population is sampled and then given to the hybridization module.

4.5. The CyberGel Module. The CyberGel module receives a sampled population and displays the nucleic acid species contained therein in the format of an electrophoretic gel. This enables a researcher to examine the contents of a population using an output modality that offers the benefits of familiarity and comparability with laboratory-produced gels. Furthermore, it permits the researcher to examine annotations associated with bands in the gel, which lends it a comparative advantage over laboratory gels. For example, for each band in the CyberGel, the researcher can learn the exact contents of the band: what strand species (or combination of species) is present in the band, what its sequence is, and how the strand was formed (courtesy of its history tag). Though the CYBERCYCLER maintains product concentrations for each strand in a population, the CyberGel module currently displays uniformly-sized bands. A better implementation would render the bands with a size and intensity proportional to the concentration of the corresponding strand species.

5. Using the CYBERCYCLER

We now turn to a discussion of how we use the CYBERCYCLER to predict the products that are formed during the operation of our unary counter. For verification purposes, we have also used the CYBERCYCLER to examine standard PCR, assembly PCR, and library assembly with ligase (as in Adleman's Hamiltonian Path experiment [**1**]), but those results are not included here.

The initial population in our CYBERCYCLER simulation consists of all the pieces of DNA necessary for implementing the programmed mutagenic unary counter (for a more detailed description of this counter see [**12, 14, 13**]). We radio-label one of the mutagenic rule strands (length 21bp) and then pass the population of five strands to the CYBERCYCLER. The CYBERCYCLER generates an extensive textual description of the products, consisting of the complete population, concentration, sequence content, bindings, reactions, and strand histories after each cycle. The CyberGel module creates a more intuitive graphical representation of the products.

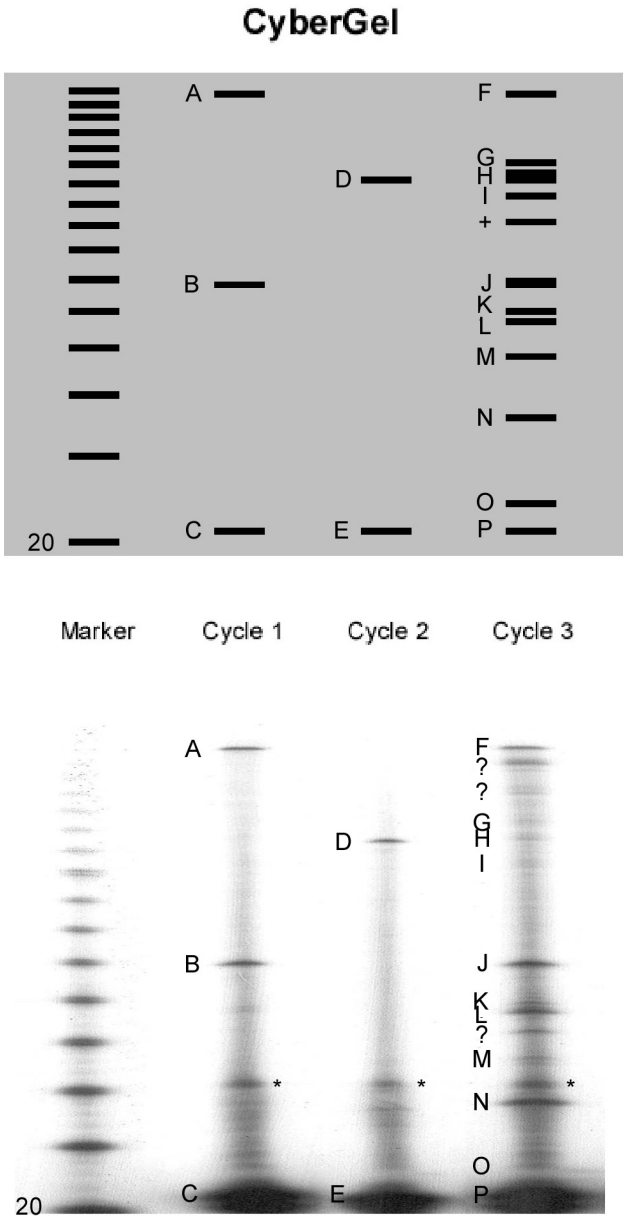

FIGURE 3. *The CyberGel output of a* CYBERCYCLER *simulation run at the same conditions as a set of reactions carried out in the laboratory. Bands in the CyberGel are shown uniformly-sized, independent of concentration. Bands labeled A through P correspond in each gel, while bands labeled with a + or ? are discrepancies. Bands labeled with a * are artifactual and can be discounted.*

In Figure 3, we present the CyberGel output of a CYBERCYCLER simulation run at the same experimental conditions as a set of reactions carried out in the laboratory. We see that the CYBERCYCLER is quite good at predicting and interpreting bands appearing in the laboratory gel. In particular, each of the bands labeled A through P in the laboratory gel has a corresponding band in the CyberGel.

The 42bp band in each cycle of the laboratory gel (labeled with a *) is an artifact and can be discounted. We have observed a band of exactly twice the length of the radio-labeled rule strand in hundreds of unary counter experiments. While the exact origin of this band is not known, it has been reproducibly observed by other researchers as well.

Nevertheless, a few discrepancies between the two gels still remain. One band in the CyberGel (labeled with a +) is not present in the laboratory gel. This is likely due to our conservative modeling: while the product represented by this band might potentially be able to form, it may not be at all likely.

Additionally, three third cycle bands in the laboratory gel (labeled with a ?) are missing in the CyberGel. We believe these bands represent strands that are shortened due to monomeric secondary structure in the unary counter template molecule. They are not predicted by the CYBERCYCLER because our hybridization module does not currently screen for monomeric secondary structure within the population.

Despite these discrepancies, one powerful feature of the CYBERCYCLER is its ability to track the histories of strands throughout the reaction. For example, if we examine the band in the CyberGel that did not appear in the laboratory gel (labeled with a +), we can see that it is 92bp long and is formed when strand C binds to strand D and is extended until it is blocked by strand B, to which it is ligated.

6. Specification of the General Framework

To provide a concrete specification of the framework developed in Section 3, we have developed Java interfaces for populations, population transformations, structures, and sequences, along with abstract classes implementing each of those interfaces. Furthermore, we have created a number of classes extending the abstract classes so that other researchers can begin to use these interfaces with minimal initial investment of time. Classes implementing these interfaces can be composed together as long as they adhere to the interface specification, enabling classes written at disparate locations to be distributed and used elsewhere. A complete description of the Java interfaces and classes, as well as all the source code for this specification are available from *http://psrg.lcs.mit.edu/biosim/*.

7. Future Work

We continue to develop the biological simulator specification and are in the process of creating more transformation modules. The CYBERCYCLER was originally written in C++ and its transformation modules are in the process of being converted to Java classes. As they are completed, they will be posted to *http://psrg.lcs.mit.edu/biosim/*.

Our modules could be extended as needed for different simulation applications. For example, the hybridization module might be re-implemented to check for hairpins and other monomeric secondary structures.

Our modules could also be supplemented as needed by implementing additional transformation modules. Modules that seem straightforward to implement include split, merge, radio-label (phosphatase and kinase), filter, immobilize, restriction cut, transcribe, reverse transcribe, and protein translate.

This large collection of modules could then be composed to represent reactions like PCR, RT-PCR, LCR, library construction, inserting a fragment of DNA into a plasmid vector, sticker bit-setting, affinity purification, etc.

An integrated simulation desktop environment would simplify tool-building immensely. A simulation desktop environment would permit a researcher to load a set of available transformation modules into his tool box, and then draw a flowchart-like representation of the experiment he wished to simulate, inserting the desired modules from his tool box as needed.

8. Conclusion

We believe our general framework for biological simulation is a useful one for a variety of reasons. Its modular design allows for code reuse and distributed code development, enables researchers to compose modules simulating arbitrarily complex sequences of reactions in the laboratory, provides an easy way for researchers to tailor modules to their desired precision or update modules as more data become available, and establishes a standard interface for future modules, yet to be developed.

The presence of taggable data is also useful in that it permits the modules to track various aspects of the reactions and provide the researcher with more information about what is happening to the various species during each reaction, much more than could be gleaned from laboratory experimentation and with dramatically less work. For instance, we implemented history tags to track when and how strands are formed during the thermocycling process, and radio-label tags to assist in generating an appropriate CyberGel output image.

We are eager to implement new modules and see what modules other groups decide to implement or improve. The collaborative possibilities of this simulation venture are extremely exciting.

Acknowledgments. The authors wish to thank Julia Khodor for her helpful comments and suggestions, and also for the provision of the electrophoretic gel displayed in Figure 3. Hartemink also gratefully acknowledges the support of the National Science Foundation.

References

[1] Adleman, L. (1994) *Science* **266**, 1021-1024.
[2] Advanced Gene Computing Technologies, Inc. *http://www.hybsimulator.com/*.
[3] Allawi, H. T. & SantaLucia, J. (1997) *Biochemistry* **36**, 10581-10594.
[4] Allawi, H. T. & SantaLucia, J. (1998) *Biochemistry* **37**, 2170-2179.
[5] Allawi, H. T. & SantaLucia, J. (1998) *Biochemistry* **37**, 9435-9444.
[6] Allawi, H. T. & SantaLucia, J. (1998) *Nucleic Acids Res.* **26**, 2694-2701.
[7] Anato, V. P. & Tinoco, I. (1992) *Nucleic Acids Res.* **20**, 819-824.
[8] Breslauer, K., Frank, R., Blöcker, H. & Marky, L. (1986) *Proc. Natl. Acad. Sci. USA* **83**, 3746-3750.
[9] Engels, W. R. (1993) *Trends in Biochemical Sciences* **18**, 448-450.
[10] Hagiya, M. *http://nicosia.is.s.u-tokyo.ac.jp/MCP/eng/index.html*.
[11] Hammes, G. (1978) *Principles of Chemical Kinetics*. Academic Press, New York.

[12] Hartemink, A. & Gifford, D. (1997) Thermodynamic Simulation of Deoxyoligonucleotide Hybridization for DNA Computation. *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers*, June.
[13] Hartemink, A., Gifford, D. & Khodor, J. (1998) Automated Constraint-Based Nucleotide Sequence Selection for DNA Computing. *Proceedings of the 4rd DIMACS Workshop on DNA Based Computers*, June.
[14] Khodor, J. & Gifford, D. (1998) Design and Implementation of Computational Systems Based on Programmed Mutagenesis. *Proceedings of 4th DIMACS Workshop on DNA Based Computers*, June.
[15] Molecular Biology Insights, Inc. *http://oligo.net/*.
[16] Petruska, J., Goodman, M. F., Boosalis, M. S., Sowers, L. C., Cheong, C. & Tinoco, I. (1988) *Proc. Natl. Acad. Sci. USA* **85**, 6252-6256.
[17] Peyret, N., Anada, P., Allawi, H. T. & SantaLucia, J. (1999) *Biochemistry* **38**, 3468-3477.
[18] Peyret, N. & SantaLucia, J. *http://sun2.science.wayne.edu/~jslsun2/hyther.htm*.
[19] Pritchard, C. E. & Southern, E. M. (1997) *Nucleic Acids Res.* **25**, 3403-3407.
[20] Quartin, R. & Wetmur, J. (1989) *Biochemistry* **28**, 1040-1047.
[21] Reif, J. H. *http://bmc.cs.duke.edu/*.
[22] Rubin, E. & Levy, A. (1996) *Nucleic Acids Res.* **24**, 3538-3545.
[23] SantaLucia, J. (1998) *Proc. Natl. Acad. Sci. USA* **95**, 1460-1465.
[24] SantaLucia, J., Allawi, H. & Seneviratne, P. A. (1996) *Biochemistry* **35**, 3555-3562.
[25] Steinfeld, J., Francisco, J. & Hase, W. (1989) *Chemical Kinetics and Dynamics*. Prentice-Hall, Englewood Cliffs, New Jersey.
[26] Sugimoto, N., Nakano, S., Yoneyama, M. & Honda, K. (1996) *Nucleic Acids Res.* **24**, 4501-4505.
[27] Werntges, H., Steger, G., Riesner, D. & Fritz, H. (1986) *Nucleic Acids Res.* **14**, 3773-3790.
[28] Weston, R. & Schwartz, H. (1972) *Chemical Kinetics*. Prentice-Hall, Englewood Cliffs, New Jersey.
[29] Wetmur, J. (1991) *Crit. Rev. in Biochem. and Mol. Biol.* **26**, 227-259.
[30] Zenkova, M. A. & Karpove, G. G. (1993) *Uspekhi Khimii* **62**, 414-435.
[31] Zuker, M., Mathews, D. H. & Turner, D. H. (1999) *Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide in RNA Biochemistry and Biotechnology*. J. Barciszewski & B. F. C. Clark, eds. NATO ASI Series, Kluwer Academic Publishers.

Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139
E-mail address: amink@mit.edu

Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139
E-mail address: tarjei@mit.edu

Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139
E-mail address: gifford@mit.edu