Experimental DNA computing

Experimental DNA computing

PROEFSCHRIFT

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van de Rector Magnificus Dr. D.D. Breimer,
hoogleraar in de faculteit der Wiskunde en
Natuurwetenschappen en die der Geneeskunde,
volgens besluit van het College voor Promoties
te verdedigen op woensdag 23 februari 2005
klokke 14.15 uur

door

Christiaan Victor Henkel
geboren te 's-Gravenhage in 1975

Promotiecommissie

Prof. dr. Herman Spaink • *promotor*

Prof. dr. Grzegorz Rozenberg • *promotor*

Prof. dr. Thomas Bäck • *promotor*

Prof. dr. Tom Head (Binghamton University) • *referent*

Prof. dr. Joost Kok

Prof. dr. Eddy van der Meijden

Dr. ir. Fons Verbeek

Het in dit proefschrift beschreven onderzoek is gefinancierd door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO), gebied Exacte Wetenschappen

ISBN 90 9019081 3

Contents

1

Introduction to experimental DNA computing

Abstract

Living systems compute, but in ways that are often hardly recognizable as such. DNA computing is an emerging field of investigation which attempts to harness the information processing power present in biology to perform formal computations.

This chapter presents the backgrounds of biological computing, followed by the motivations behind the construction of molecular computers, and DNA based computers in particular. Potential applications are discussed, and an overview of experimental progress is given. Finally, the research described in this thesis is introduced.

Natural computing

Information and communication are ubiquitous in biology. The most obvious example from molecular biology is genetic information, which is stored, transferred, replicated, translated and recombined. On a biochemical level, all proteins and nucleic acids perform complicated pattern recognition tasks, and signal transduction and processing is central to cell biology. On higher levels, the human brain is in many ways still the supreme information processor, and evolutionary mechanisms are unmatched in the complex task of adapting to an environment.

Yet officially, computer science is the discipline that deals with information and its processing. Apart from being enormously successful in the construction of electronic computers, this field has provided fundamental insights into information processing.

The artificial dichotomy between these sciences of information is resolved by the emerging field of natural computing. This recent scientific discipline explores both nature inspired computing and actual computation taking place in nature (Rozenberg & Spaink, 2002). Amongst its subjects are established problem solving strategies, such as evolutionary algorithms and neural networks. Evolutionary computation borrows from evolution by natural selection in order to deal with optimization problems (Eiben & Smith, 2003). Candidate solutions are subjected to (*in silico*) mutation, recombination, selection and breeding. Neural computation is inspired by animal nervous systems and uses networks of simulated neurons for various computational tasks, such as pattern recognition and data classification. Both approaches are particularly useful when the computational problem considered does not allow for a more traditional approach, for instance when knowledge of problem and solution structure is limited.

Natural computing also encompasses nascent branches such as molecular

and quantum computing (Bennett & DiVincenzo, 2000), both of which aim at the use of more or less natural structures and processes for the implementation of computations. (Of course, all computation is ultimately dependent on physical structures; Bennett & Landauer, 1985. Natural computing is therefore predominantly concerned with non-traditional hardware.)

The hopes of natural computing are not only to advance those subjects, but also to gain insight into the character of computation itself (MacLennan, 2003), and to understand natural processes better by assessing them in the light of formal computation. The investigations into gene assembly in ciliate protozoa serve as an example of the latter (Landweber *et al.*, 2000; Prescott & Rozenberg, 2002; Ehrenfeucht *et al.*, 2004).

Molecular computers

There are several reasons to pursue the construction of molecular scale computers. One of the most obvious is just following the trend of miniaturization, advocated already by Feynman (1959), which has been present in microelectronics over the last four decades.

This tendency was first recognized by Moore (1965), and is now known as Moore's law. An economic principle rather than a law of nature, it states that transistor sizes will continue to shrink so the space they occupy halves roughly every two to three years (figure 1). This leads to the possibility of increasingly complex logic chips, higher capacity memory chips and lower switching times. Current lithographic technology produces microchips with defining details of only 90 nanometres (meaning that some parts are of even smaller dimensions). If Moore's law is made to hold much longer, transistor sizes will eventually reach the scale of individual molecules and atoms. It is far from certain that it will be possible to construct integrated circuits of silicon-based solid state transistors using familiar 'top-down' technology (using light-directed lithography), and if so, whether they will be functional (Packan, 1999; Lundstrom, 2003). Both quantum phenomena and increasing heat generation appear prohibitive for the persistence of the trend.

A recent technology which hopes to deal with these problems is molecular electronics, which tries to replace conventional electronic elements (such as semiconductor transistors and wires) with molecules (Tour, 2000). Most of the components considered are organic molecules or carbon nanotubes (Bachtold *et al.*, 2001), and even biological macromolecules are promising, as in the proposed light-addressable rhodopsin memory (Birge *et al.*, 1999). Manufacturing techniques for molecular electronics and nanotechnology are generally 'bottom-up', in which individual components arrange themselves through local interactions

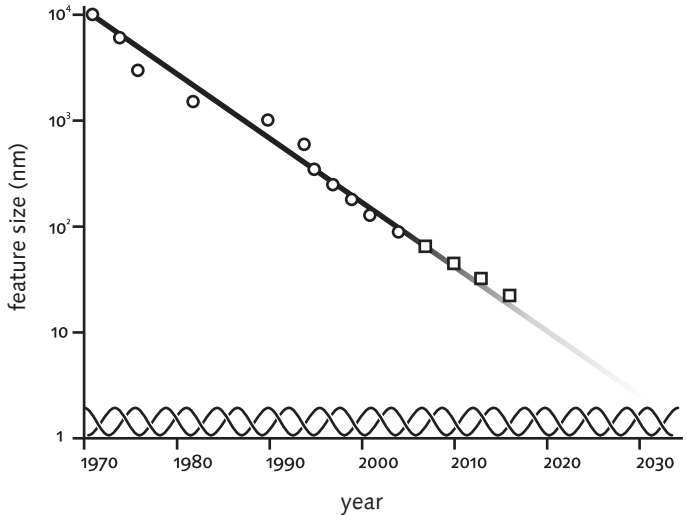

Figure 1. Defining feature sizes produced in mass production silicon lithography. Circles indicate production processes employed for microprocessor production (Intel, 2004), and squares represent technology projections up to 2016 by the International Technology Roadmap for Semiconductors (2003). The line indicates the Moore's law trend of miniaturization. Extrapolation predicts molecular scale transistors by the 2030s, illustrated here with the 2 nm helix dimensions of DNA.

(self-assembly). However, the general functionality that is aimed for is still very similar to solid state electronics: elements should act as switches, pass electrons, and have permanent and definable contacts with other components.

Another reason to pursue the construction of molecular scale computing devices is their scale. Some applications may simply call for very tiny, but not necessarily powerful computers.

Finally, molecules may provide ways to implement completely different computing architectures. All current computers are still largely based on variants of the traditional von Neumann architecture (Burks *et al.*, 1946): a single logic processing unit, a single sequentially addressed memory, a control unit and a user interface, and consequences such as the distinction between hardware and software. While this design has proved hugely successful, it is not necessarily synonymous with a computer, and other designs may cover computing needs that are hard to achieve using conventional means. This notion can be illustrated with a trade-off: 'A system cannot at the same time be effectively programmable, amenable to evolution by variation and selection, and computationally efficient' (Conrad, 1985). This certainly seems plausible when one compares von Neumann computers to biological systems. The former is multi-purpose, and very programmable. However, its use of space, time and energy resources is quite inef-

ficient. Biological systems are lacking in programmability and general control, but through superior adaptability are able to efficiently solve complex problems. Both systems are extremes in this trade-off, and if it holds, it is conceivable that some middle ground exists for powerful and practical molecular computers.

Design principles of biomolecular computers

The behaviour of molecules under normal (for instance physiological) conditions is drastically different from relatively macroscopic components, such as solid state transistors. For example, one of the greatest implementation challenges of molecular electronics is just to keep parts from wandering aimlessly through circuits by diffusion. However, random diffusion and other molecular processes may be a blessing in disguise, since they hold considerable computational potential.

Molecules can contain and process information in many ways, for example through reactive groups, conformational changes, electron transfer or optical properties. Operations on such information are performed by the interactions of molecules. The basic operations for biological macromolecules can be described as shape or pattern recognition, with subsequent conformational change and often catalysis. Suitably complex molecules have many more states than just the binary 'on' and 'off', and the exploration of complementary shape is actually a highly parallel optimization procedure on an energy landscape. Plausible timescales for these operations to occur (switching times) are on the microsecond scale, although electron transfer and optical switching can be much faster. Gigahertz molecular computers based on allosteric mechanisms are therefore not realistic – however, what they lack in speed molecules can make up for in numbers.

Molecular computers that operate through shape and pattern recognition are necessarily three-dimensional in nature, where components can move freely and engage in actions in all directions. This is a radical departure from the rigid integrated circuits of electronics, where physical interactions between transistors are fixed. Predetermined structures are also typical for molecular electronics, although their organization is not intrinsically limited to two dimensions. The paradigm of free diffusion is central to much of molecular computing, as it allows for massive parallelism. This does not only depend on the huge numbers of elements that can participate in a computation: typical molecular biology procedures use picomolar quantities, or 10^{12}–10^{14} molecules, which could all act as single simple computer processors. Because of the thermal noise molecular components experience, the search for correct interactions is thermodynamically free (this Brownian search principle can also be exploited for unconventional modes of computing, such as the implementation of reversible computation or nondeterminism; Bennett, 1982). Only the final act of (irreversible) informa-

tion transformation requires the dissipation of energy (Schneider, 1991, 1994). Biochemical information processing is usually coupled to hydrolysis of ATP to fulfil this requirement. As such, biological systems are remarkably efficient with energy: via ATP hydrolysis, 10^{19} operations per Joule can be performed, close to the theoretical limit of 3.4×10^{20} per Joule dictated by the Second Law of thermodynamics (Schneider, 1991; Adleman, 1994). This alone could be motivation enough to pursue the construction of molecular computers, as state-of-the-art silicon processors dissipate up to 100 Joule for approximately 10^{10} binary operations per second.

DNA as a substrate for computation

The advent of molecular biology has been accompanied by metaphors taken from information and computer science. Cellular order and heredity were inferred to rely on information intake ('negative entropy'; Schrödinger, 1944) and genetic information is still thought of as a 'code'. Biological regulatory systems were identified as 'microscopic cybernetics', which 'abide[s] not by Hegelian laws but, like the workings of computers, by the propositional algebra of George Boole' (Monod, 1971). Processes involving nucleic acids, such as transcription and translation, are reminiscent of the tape operations of a Turing machine, the dominant model of universal computing (Bennett, 1973; Adleman, 1998). Given such precedents, the idea of artificial molecular biological computers is an almost inevitable development.

Early suggestions on the construction of biomolecular computers always emphasized protein components (Drexler, 1981; Conrad, 1985, 1992; Bray, 1995). Still, nucleic acids appear to be a natural choice for the construction of molecular computers. Not only are they amongst the most intensively studied molecules, and very well characterized in comparison to other complex macromolecules, but they also already show support for information technology through their roles in genetics.

DNA characteristics suitable for computation

Study of DNA structure and function has yielded many insights into attributes that can in retrospect be linked to computational qualities. Some of the characteristics that in theory make DNA a good computing molecule are given here, together with other, more practical considerations.

Information storage. The linear sequence of nucleotides in DNA is a comparatively straightforward way to encode information. The system is not unlike the binary representation of data in conventional computers, except that for every

position (nucleotide or basepair), there are four different possibilities instead of just 1 and 0. The information content of a single nucleotide position is then $\log_2 4 = 2$ bits.

Pattern recognition. The principal logic of DNA is in its pattern recognition abilities, or hybridization. Given permitting conditions, complementary single strands of DNA will hybridize, or anneal, to form a double helical molecule. The process is reversible: altered conditions, most notably elevated temperatures, can overcome the basepairing energies. 'Melting' a DNA helix results in the return of the constituent single strands to a random coil state. Hybridization is in essence a complicated molecular search operation, with intricate kinetics. For computing purposes however (as for most of molecular biology), the process can be described by and predicted with simple models and empirical formulas (Wetmur, 1991; SantaLucia, 1998). As hybridization is dependent on nucleotide sequence, it allows for programmable interactions between molecules.

Solubility. Molecular search operations are dependent on random diffusion of molecules through a suitable solvent. The sugar-phosphate backbone of nucleic acids confers high solubility in water upon the otherwise hydrophobic nucleobase information.

Basic modification. In order to compute, the information in DNA must be processed. An extensive molecular toolkit is available to manipulate this information. Possible operations can involve only nucleic acid (for example, denaturation and annealing), or take advantage of the many DNA modifying enzymes available. The most interesting are probably the restriction endonucleases, which act on specific molecular information. Other possibilities include polymerases, ligases, exonucleases and methylases. More comprehensive treatment of these operations in the context of DNA computing can be found in Păun *et al.* (1998).

Visualizing results. A multitude of analytical techniques is available to visualize the information present in DNA. Examples are gel electrophoresis, nucleotide sequencing and array hybridization. These can be employed to detect the output signals of DNA computations. Also of interest are amplification techniques (polymerase chain reaction, rolling circle amplification) that may be used to boost molecular signals.

Availability. Natural DNA is ubiquitous and readily isolated and purified. This is probably not the best source of computing DNA, as this use imposes many constraints on nucleotide sequences. Chemical synthesis of DNA is another potential source. Nanomolar quantities of DNA up to several hundred nucleotides are routinely produced at low cost. Larger stretches of DNA can be produced by concatenation of synthesized oligonucleotides; however, this is a cumbersome and error-prone process.

Stability. Although any molecule will eventually decay, DNA is stable compared to other biological macromolecules. Due to the lack of a 2'-hydroxyl group,

the phosphodiester bond is far more stable in DNA (with an estimated half-life of 45000 years for a single linkage, under physiological conditions; Radzicka & Wolfenden, 1995) than in RNA (half-life nine years; Li & Breaker, 1999). DNA is more sensitive than RNA to spontaneous depurination and subsequent backbone cleavage, although the reaction rates are still low (half-life >2000 years; Lindahl, 1993). The peptide bond in proteins has a half-life of the order of 250 years (Smith & Hansen, 1998). Storage conditions strongly affect these parameters, for example partially dehydrated DNA can survive for thousands of years. It would appear that such timescales allow for meaningful computations. Still, in designing a DNA based computer one should keep in mind that the molecules are constantly degrading. If this becomes a problem, a solution might consist of including multiple, redundant copies of every molecule. Alternatively, one could consider including cellular DNA maintenance and repair mechanisms in the system.

Algorithmic implementation. DNA has an excellent reputation as a major component of natural molecular computing systems, molecular biologists even routinely 'program' cells through genetic engineering. Furthermore, the solution to molecular design problems through *in vitro* evolution is already very close to computation (Wilson & Szostak, 1999; Joyce, 2004). Other (natural) processes also allow for computational interpretation. It would therefore appear feasible to use DNA in man-made computers.

Integration with biology. Finally, an interesting niche for molecular computers may be to process data from molecular systems, the most interesting of those being living systems. It then makes sense to construct molecular computers from components compatible with organisms. In addition, such components may function as an interface between computers (of any architecture and composition) and life.

The first synthetic DNA computer

The first example of an artificial DNA based computing system was presented a decade ago (Adleman, 1994). This special purpose DNA computer solved a small instance of a hard computational problem, the Hamiltonian Path Problem (HPP). Given a graph with nodes (vertices) and connections (edges), this problem asks for a path with fixed start and end nodes, that visits every node exactly once (figure 2a). To solve this problem, every node was encoded as a 20 nucleotide oligomer. Connections were encoded as 20-mers, with the first 10 nucleotides complementary to the last 10 nucleotides of the start node, and the last 10 complementary to the first 10 of the end node. This way, a connection oligonucleotide can bring the two nodes it connects together by acting as a splint (figure 2b).

By mixing and ligating oligomers corresponding to every node and every connection, concatenates are formed that represent paths through the network

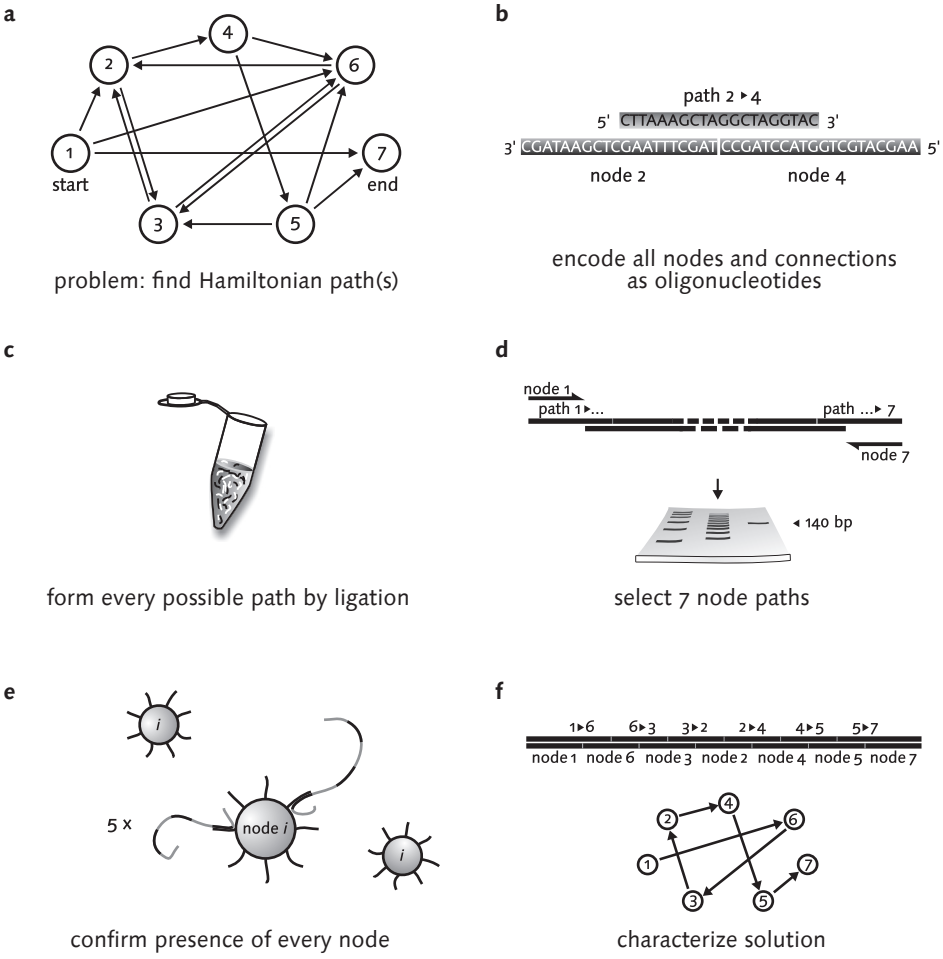

Figure 2. DNA solution to a Hamiltonian Path Problem instance (Adleman, 1994). **a** The graph used. **b** Encoding strategy. The oligonucleotides encoding two nodes (5'→3') and the edge connecting (3'→5') them are shown. **c** Mixing all seven nodes and 14 paths results in the ligation of all possible paths through the graph. Nodes 1 and 7 are not strictly necessary, as their presence in the final paths can be encoded by the splint oligos alone. Incorporation promotes the formation of paths both entering and leaving these nodes. **d** Selection of paths of correct length. The ligation product was amplified by PCR, using the oligonucleotide encoding node 7 and the complement of node 1 as primers. After gel electrophoresis, products of 140 bp (seven nodes) long were selected. **e** This product was denatured, and only those strands containing node 2 were selected using beads coated with the complement of node 2. This procedure was repeated for nodes 3 to 6. **f** Output of the computation. Presence and position in the path of every node was verified by PCR using complements of this node and the oligo for node 1 as primers. Only the correct Hamiltonian path was retrieved from the ligation mixture.

(figure 2c). Not just any path through the graph is a solution to the problem. Random ligation will form many paths that do not meet the conditions set. Therefore, several selection steps are required (figure 2d, e): first, use PCR to select only those paths that start and end at the right node; then, keep only paths of correct length (seven nodes times 20 nucleotides); and finally, confirm the presence of every node sequence (using affinity separation). If any DNA remains after the last separation, this must correspond to a Hamiltonian path. Experimental implementation of this protocol indeed recovered a single species of oligonucleotide, which was shown to encode the only possible Hamiltonian path through the graph (figure 2f).

From a computer science point of view, the path-construction phase of the algorithm is the most impressive. Because of the huge number of oligonucleotides used (50 picomol per species), all potential solutions are formed in parallel, in a single step, and through chance molecular encounters.

Solving hard problems as a potential application

Although the seven node problem above appears quite easy, in general, the HPP is a very hard problem. Essentially the only way to solve it is by exhaustive evaluation of all possible paths through the graph, and this number of paths increases exponentially with the size of the network. Consequently, solving a HPP on a von Neumann computer (with a single processing unit) requires an amount of time that grows exponentially in response to a linear increase in input size. Such problems then quickly become infeasible to solve (intractable).

The HPP is a representative of a whole group of problems with similar scaling behaviour: the class of non-deterministic polynomial problems, or NP. The name reflects the fact that such problems can be solved on timescales bounded by a polynomial function only through guessing the solution (non-determinism) and verifying the guess. In contrast to true exponential time complexity problems, NP problems have the property that answers can be checked in polynomial time. For example, finding a Hamiltonian path is hard (takes exponential time), but confirming that the path is indeed Hamiltonian is easy (takes polynomial time).

A special subclass of NP includes problems that can be converted into one another on polynomial timescales. If an efficient (i.e. polynomial time complexity) algorithm can be found for any one of these problems, all problems in this NP-complete class can be solved efficiently. No such algorithm is known to exist, but it has not been proved not to exist either (Garey & Johnson, 1979). Figure 3a shows the relationship between various classes of computational problems, classified by complexity.

Since many NP-complete problems are economically very important (for example many scheduling and optimization problems fall into this class, in addi-

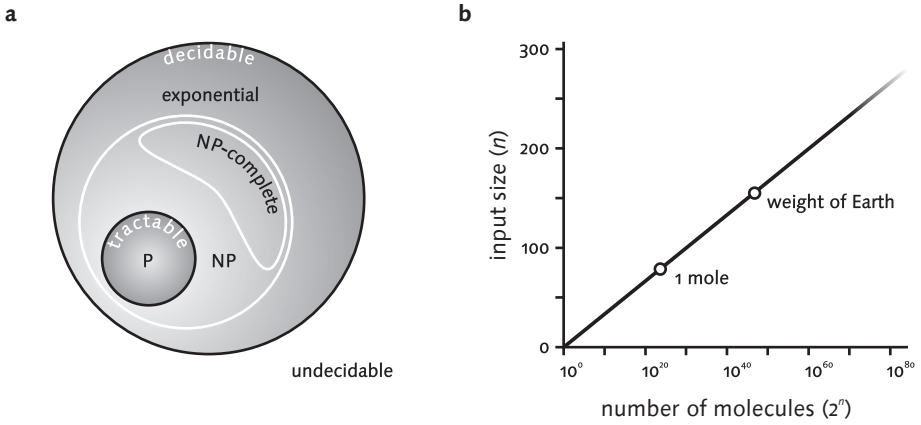

Figure 3. Computational complexity. **a** The space of algorithmic problems. Tractable problems can be solved by algorithmic means in polynomially bounded time (class P). Intractable problems require exponential amounts of time or space to arrive at a solution. Problems in NP are in practice intractable, but lower bounds on their time complexity are not known (i.e. does class P equal class NP is an open question, in fact one of the most important questions in mathematics). Answers to intractable problems can in theory still be produced by computational means. Other problems are fundamentally undecidable, and are not solvable by any algorithm. **b** Exponential complexity in practice. Shown is the behaviour of a computation with complexity 2^n for input size n. A brute force molecular algorithm has to represent every potential solution as a molecule. The number of molecules quickly becomes unreasonable for moderate input sizes (adapted from Mac Dónaill, 1996).

tion to the HPP), a method to compute their solutions efficiently would be of great value. Currently, heuristic algorithms are often used which trade time for precision, i.e. sub-optimal solutions are calculated and accepted on manageable timescales. Following Adleman (1994), it was suggested that DNA might provide a way to attack NP-complete problems (Gifford, 1994). In contrast to sequential computers, the time required to solve a HPP on a DNA computer (expressed in the number of biochemical operations) scales linearly instead of exponentially with respect to input size: for instance, doubling the number of nodes takes only twice the number of separation steps. And although DNA computing is very slow in comparison with silicon, in theory it can make up for this by the enormous parallelism that can be accommodated. Around 10^{12}–10^{14} DNA strands, each corresponding to a potential solution, can be processed in parallel.

It was quickly pointed out that computing with DNA as described above does not provide a real escape from the exponential behaviour of NP-completeness, and that time is simply being traded for space. Several articles calculate how brute force molecular computers for solving non-trivial instances of the HPP would require the weight of the Earth in nucleic acid (Hartmanis, 1995) or oc-

cupy the entire universe (Bunow, 1995; Mac Dónaill, 1996; figure 3b). However, such arguments do nothing to disqualify the application of DNA computing for NP-complete problems, they merely illustrate the intrinsic difficulty of dealing with these problems. The search spaces attainable with DNA are still vastly greater than those possible with other, more conventional means, and molecular computers may therefore yield significantly more powerful heuristic approaches.

Information storage in DNA

The parallelism provided by DNA computers is not only useful in solving intractable problems. The available search spaces might be used in the construction of molecular memories, or databases (Baum, 1995; Reif & LaBean, 2001). The basic idea is very similar to the solution of combinatorial optimization problems: every species of DNA in a molecular memory corresponds to a database entry, and queries upon the database can be executed through the same separation technologies employed in parallel DNA computing. The most remarkable advantage of such databases is again their potentially enormous size. However, such databases may also benefit from the idiosyncrasies of DNA separation technologies; query conditions may be altered to retrieve not only perfect matches, but also closely associated entries. DNA databases could also be loaded with biologically relevant data, e.g. natural (c)DNA (with or without specific address labels; Brenner *et al.*, 2000) or small peptides (Halpin & Harbury, 2004).

Data storage in DNA is a tempting idea in general. The storage capacity of nucleic acids is of the order of one bit per nm^3, vastly larger than that of conventional optical or magnetic media (the information contained in a gram of DNA is approximately 200 exabytes, which corresponds to a stack of 4.6×10^{10} DVDs, with a capacity of 4.7 gigabytes each). Readout and encoding speeds are of course extremely slow in comparison, which is not only a limitation of current sequencing technology but probably intrinsically coupled to the speed of enzymatic DNA processing. Still, DNA has been considered for very long-term data storage (Cox, 2001; Bancroft *et al.*, 2001). The rationale behind this option is the unlikelihood that DNA will ever become obsolete, as the majority of twentieth century storage media already have. In addition, DNA data degradation is slow. DNA cryptography and tagging are already more feasible (Leier *et al.*, 2000; Clelland *et al.*, 1999). Specific DNA sequences can be added to any nuclease-free material to provide an invisible marker, which can only be accessed by someone who knows what to look for (e.g. someone who possesses the proper PCR primers to amplify the message).

Other applications

Another interesting niche for DNA based computers is in bioinformatics itself: the processing of biological data. Several proposals have been put forward to analyse gene expression data using molecular computing methods (Sakakibara & Suyama, 2000; Mills, 2002). These data sets are typically very large (ideally spanning a whole transcriptome), but require only simple operations (straightforward comparisons between several samples). Best of all, they are available in molecular format. Sakakibara & Suyama (2000; see also Normile, 2002) have proposed intelligent DNA chips, which perform simple logic operations on cDNA hybridized to the array. This approach eliminates detection steps and costly data processing on conventional computers, and is therefore potentially faster and more reliable. Another approach to gene expression profiling has been proposed in which a neural network is encoded in DNA strands, with DNA concentrations corresponding to neuron strengths (Mills, 2002). Mixing of the network and a cDNA input should give a verdict on certain characteristics of the expression profile. Such a system could be used for clinical purposes (i.e. quick diagnosis on cell samples), with the added advantage of minimal human influence.

The latter approach is not concerned anymore with the parallelism provided by molecular computing, although that could serve as a signal boosting method (performing exactly the same cDNA analysis a million times). Several other applications are conceivable where simple operations on relatively few data are needed, but at the molecular scale. For example, biosensors could be constructed which perform a task similar to the molecular neural network described, but on any molecular data set. Promising candidates are (deoxy)ribozymes (Breaker, 2000, 2002), which can be efficiently programmed to act as logic switches and perform simple molecular computations (Stojanovic & Stefanovic, 2003). It is conceivable that similar components may be used for therapeutic ends, in a sort of smart gene therapy which decides on an action on the basis of cellular conditions (mRNA levels).

Finally, DNA computing may even contribute to molecular electronics. Several roles are conceivable for nucleic acids, ranging from construction material of logic gates to electric wire (the conductive qualities of DNA are debated, but probably not up to the task; Bhalla *et al.*, 2003; Armitage *et al.*, 2004). Promising is the use of DNA to guide the arrangement of other molecular electronic components through its programmable interactions, as a bottom-up manufacturing scheme (Drexler, 1981; Seeman, 1999; Braun & Keren, 2004). Several switch and wire materials have already been coupled to DNA to enable such molecular positioning (Williams *et al.*, 2002; Liu *et al.*, 2004), and DNA has been employed as a template for a carbon nanotube transistor (Keren *et al.*, 2003).

Progress in DNA computing research

The Hamilation path experiment (Adleman, 1994) initiated a whole area of research, and there have been numerous studies on DNA based computers. Reports have been published on theoretical principles, design aspects, possible algorithms, and laboratory techniques applicable in computations. Finally, there have been a number of articles describing complete nucleic acid based computations.

Theoretical studies

There has been considerable effort to formalize biological computing and subsequently assess its power (Păun *et al.*, 1998). Currently, two models are particularly popular: splicing and membrane systems, also known as H and P systems, respectively. Splicing systems (Head, 1987) are inspired by DNA recombination, and consist of DNA, restriction endonucleases and ligase. The combined action of these enzymes results in the exchange of specific DNA sequences between molecules. The possible sequences that can be generated this way are studied in the framework of formal language theory. Some variants of splicing systems are equal in computational power to a universal Turing machine: they are capable of computing any computable function.

Membrane systems consider computational structures modelled after cellular organization. They consist of nested compartments, which communicate with each other by transferring hypothetical molecules according to specific rules (Păun, 2001; Păun & Rozenberg, 2002). Such systems can also be computationally universal. For reviews of these and other theoretical models, see Păun *et al.* (1998) and Yokomori (2002).

Experimental benchmarks and innovations

The only inputs the first special purpose DNA computer could handle were instances of the Hamiltonian Path Problem (Adleman, 1994). Although any problem in the class NP-complete can in theory be transformed into a HPP, this may not always be very practical. Lipton (1995) therefore did the inverse: he adapted Adleman's design so that it could handle binary numbers, and provided an algorithm for solving problems in Boolean logic. In the Lipton architecture and others, potential solutions are represented as linear DNA bit strings, where subsequences encode the value (1 or 0: true or false) of specific bits. This design can be used to solve instances of the Satisfiability (SAT) problem, which asks for a 'satisfying' assignment of truth values to a given logical formula. Satisfaction is achieved if the formula is 'true' on a given input (assignment) of variables. For example, the formula *(**a** or **b**) and (not **a** or not **b**)* is satisfiable by the assign-

ments {***a***=*true*, ***b***=*false*} and {***a***=*false*, ***b***=*true*}, but is falsified by {***a***=*true*, ***b***=*true*} and {***a***=*false*, ***b***=*false*}. While solving this particular example is trivial, the general form of the SAT problem is NP-complete (Garey & Johnson, 1979).

The statements on variables are called literals, and can be either the variable or its negation. SAT problems are usually expressed in conjunctive normal form (CNF), which entails the disjunction (separation by logical *or*) of literals in clauses that are themselves connected by the *and* operation. The above example formula, a conjunction of two clauses, is in CNF.

The most common form of SAT is 3SAT, which requires that every clause of a CNF formula contain exactly three literals. Other forms of SAT, like any other NP-complete problem, can be reduced to 3SAT in polynomial time. (The above example is easy, not only in the trivial sense because it is short, but also in the technical sense because SAT problems with at most two literals per clause *are* solvable in polynomial time).

Following the HPP and SAT, many other architectures and algorithms to attack NP-complete problems have been proposed, only a few of which come with experimental evidence (i.e. have been implemented in molecular biological laboratories). Tables 1 and 2 list probably all DNA computations on NP-complete problem instances published to date, with table 1 summarizing the computational aspects of these implementations and table 2 the technical side. To keep the list manageable, only those experiments in which a complete computation was carried out are listed. Several DNA computer architectures are illustrated in figure 4.

All of these DNA implementations are of the 'proof of principle' scale. They do not pose any threat to silicon based computers, and are not necessarily meant to. The main accomplishment of these experiments is technical, with computations to NP-complete problem instances serving as benchmarks, to evaluate methods that have potentially much wider application areas than powerful computers. The synthetic nature of these benchmarks requires unprecedented control over complex mixtures of molecules, which is demonstrated by the synthesis of combinatorial libraries, low error parallel operations and highly sensitive analysis. Still, progress is apparent on the computational side, for example the computation by Braich *et al.* (2002) is already beyond the reasonable capacity of human trial and error computing. Another large computation (10 variable, 43 clause 3SAT) has been reported (Nakajima *et al.*, 2002), however experimental evidence has not yet been published.

The leading library synthesis methods are splint ligation (as implemented by Adleman, 1994; see figure 2), direct chemical synthesis and parallel overlap assembly, a method adapted from *in vitro* evolution (Kaplan *et al.*, 1997). The aqueous/plasmid methodology of Head (2000; figure 4) takes a different path, and starts with a single species of plasmid that can be modified by split and pool

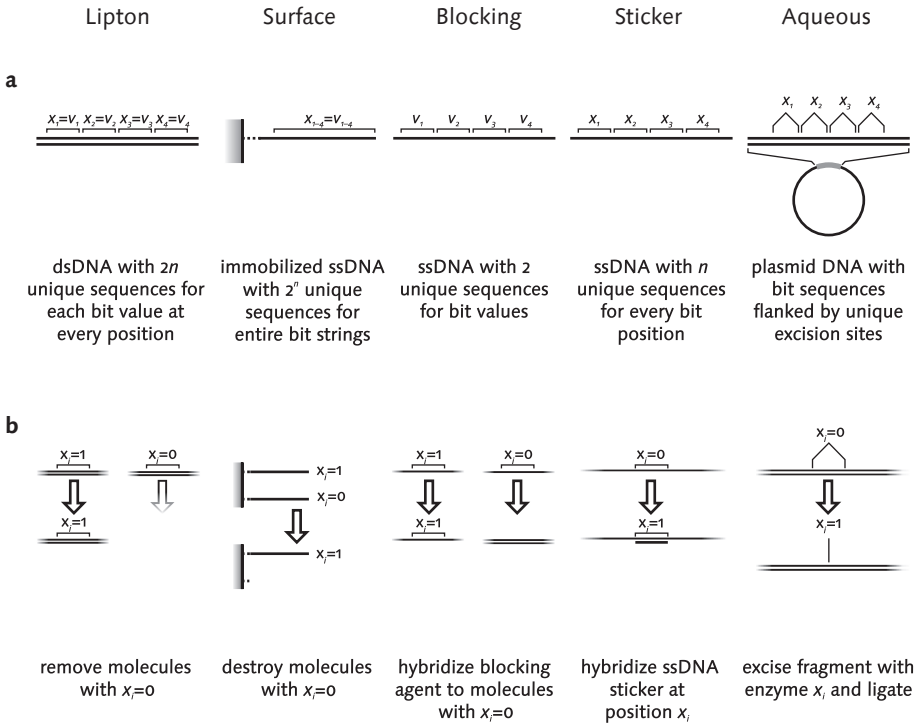

Figure 4. Architectures for DNA based parallel search algorithms. Shown are **a** representations of bit strings (values v_i for bits x_i, $1 < i < n$, here for n=4) and **b** operation on those bits (setting bit x_i to 1) in five major models: the method of Lipton (1995), the surface based approach (Smith *et al.*, 1998; Liu *et al.*, 2000), the blocking algorithm (Rozenberg & Spaink, 2003; chapters 2 and 3 of this thesis), the sticker model (Roweis *et al.*, 1998) and aqueous (plasmid) computing (Head, 2000; Head *et al.*, 2000; chapters 4 and 5). All except the sticker model have been implemented experimentally (because of its local DNA melting requirement, sticker based computation in its original form is very difficult to execute – however a strategy employing catalyst strands might be feasible; Yurke *et al.*, 2000). The five models fall into two categories. Lipton, surface and blocking start with a mixture of all possible bit strings (basically a read-only memory) and set bit values by discarding those molecules that have another value for that bit. In the sticker and aqueous algorithms it is possible to reversibly alter the value of a bit in every molecule. These models start with a single species of molecule, typically with every bit set to zero. Bit operations on subsets of this molecular random access memory generate a library, which is then searched using other techniques.

removal of subsequences. Like the sticker architecture, the aqueous method relies on a random access memory (RAM), whereas the other designs in figure 4 employ a kind of read-only memory (ROM).

Table 1. Parallel search DNA computations

Reference	Problem	Dimensions	Solved for
Adleman (1994) [a]	Directed Hamiltonian Path	*n* vertices, *m* edges	*n*=7, *m*=14
Ouyang *et al.* (1997)	Maximal Clique	*n* vertices, *m* edges	*n*=6, *m*=11
Aoi *et al.* (1998)	Knapsack (Subset Sum)	*n* items	*n*=3
Yoshida & Suyama (2000)	3-Satisfiability	*n* variables, *m* clauses	*n*=4, *m*=10 [b]
Faulhammer *et al.* (2000)	Satisfiability	*n* variables, *m* clauses	*n*=9, *m*=5
Head *et al.* (2000)	Maximum Independent Set	*n* vertices, *m* edges	*n*=6, *m*=4
Liu *et al.* (2000)	Satisfiability	*n* variables, *m* clauses	*n*=4, *m*=4
Pirrung *et al.* (2000)	3-Satisfiability	*n* variables, *m* clauses	*n*=3, *m*=6
Sakamoto *et al.* (2000)	3-Satisfiability	*n* variables, *m* clauses	*n*=6, *m*=10
Wang *et al.* (2001)	Satisfiability	*n* variables, *m* clauses	*n*=4, *m*=5
Braich *et al.* (2002) [c]	3-Satisfiability	*n* variables, *m* clauses	*n*=20, *m*=24
Head *et al.* (2002a)	Satisfiability	*n* variables, *m* clauses	*n*=3, *m*=4
Head *et al.* (2002b)	Maximum Independent Set	*n* vertices, *m* edges	*n*=8, *m*=8
Liu *et al.* (2002)	Graph Coloring	*n* vertices, *m* edges	*n*=6, *m*=12
Lee *et al.* (2003, 2004)	Travelling Salesman	*n* vertices, *m* edges	*n*=7, *m*=23
Takenaka & Hashimoto (2003)	3-Satisfiability	*n* variables, *m* clauses	*n*=5, *m*=11 [d]
Chapters 2 & 3	3-Satisfiability	*n* variables, *m* clauses	*n*=4, *m*=4
Chapter 4	Minimal Dominating Set	*n* vertices, *m* edges	*n*=6, *m*=5
Chapter 5	Knapsack	*n* items	*n*=7

a Repeated for *n*=8, *m*=14 by Lee *et al.* (1999)
b *n*=10, *m*=43 has been claimed using similar methods (Nakajima *et al.*, 2002)
c Also solved for *n*=6, *m*=11 (Braich *et al.*, 2001)
d For only 3 variables out of 5 a value was determined

Initial work	Data pool generation: Initial sp.	steps	final sp.	Selection [e]: species	steps
	$n+m$	1	unknown	n	n
specify complementary graph: $\frac{1}{2}(n^2-n)$	$2n$	1	2^n	n enzymes	$\frac{1}{2}(n^2-n)-m$
	$3n-1$	1	2^n	2	1
formula reordering if necessary: m	$4+4n$	$n-2$	$\leq 2^n$		m
determine m falsifying conditions	[f]	n	2^n	$2n$	m
	1	m	$\leq 2^m$	n enzymes	1
evaluation of 2^n solutions	[f]	2^n	2^n	2^n	m
determine m falsifying conditions	[f]	n	2^n	$2n$	m
	$3m$	1	2^m		1
evaluation of 2^n solutions	[f]	2^n	2^n	2^n	m
determine m falsifying conditions	[f]	n	2^n	$2n$	m
	1	n	2^n	$2n$ enzymes	m
	1	n	2^n	n enzymes	1
	n^2	1	n^n	n^2 enzymes	nm
	$n+m$	1	$<\sum_{x=1}^{10} x!$ [g]	n	n
determine m falsifying conditions	[f]	n	2^n	$m2^{n-3}$	1
determine m falsifying conditions	[f]	2^n	2^n	$m \leq x \leq m2^{n-3}$	1 or m
nm neighbourhood evaluations	1	$\leq n$	$\leq 2^n$	n enzymes	1
	1	n	2^n	n enzymes	1

e For several computations, the selection and library generation phases are not as discrete as suggested here – see text for details
f Direct chemical synthesis of library
g Estimate

Table 2. Technical aspects of DNA computations

Reference	Library generation	Molecule	Selection criteria
Adleman (1994)	splint ligation	dsDNA	length, subsequence
Ouyang *et al.* (1997)	overlap assembly	dsDNA	subsequence, length
Aoi *et al.* (1998)	splint ligation	dsDNA	length
Yoshida & Suyama (2000)	splint ligation	ssDNA	subsequence
Faulhammer *et al.* (2000)	combinatorial chemical synthesis	ssRNA	subsequence
Head *et al.* (2000)	combinatorial subsequence removal	plasmid DNA	length
Liu *et al.* (2000)	chemical synthesis	immobilized ssDNA	entire sequence
Pirrung *et al.* (2000)	combinatorial chemical synthesis	immobilized ssDNA	subsequence (single nucleotide)
Sakamoto *et al.* (2000)	ligation	ssDNA	subsequence
Wang *et al.* (2001)	chemical synthesis	immobilized ssDNA	entire sequence
Braich *et al.* (2002)	combinatorial chemical synthesis	ssDNA	subsequence
Head *et al.* (2002a)	combinatorial restriction site removal	plasmid DNA	subsequence
Head *et al.* (2002b)	combinatorial restriction site removal	plasmid DNA	length
Liu *et al.* (2002) [a]	overlap assembly	dsDNA	length
Lee *et al.* (2003, 2004)	splint ligation	dsDNA	length, subsequence
Takenaka & Hashimoto (2003) [a]	combinatorial chemical synthesis	immobilized ssDNA	subsequence
Chapters 2 & 3	chemical synthesis	ssDNA	subsequence
Chapter 4	combinatorial subsequence removal	plasmid DNA, protein	length, protein mass
Chapter 5	combinatorial subsequence removal	plasmid DNA, protein	length, protein mass

a These articles contain very limited experimental information

Selection technology	Readout technology	Error rate [b]	Readout [b] ambiguity	Architecture
electrophoresis, PCR, subseq. selection (beads)	PCR, electrophoresis	0		
restriction endonuclease digestion, electrophoresis	cloning, sequencing	0		
PCR, electrophoresis	cloning, sequencing	0		
subsequence selection (beads)	PCR, electrophoresis	0		Ogihara & Ray (1997)
RNase H duplex endonuclease	cloning, PCR, electrophoresis	2.3%		Lipton (1995)
electrophoresis	cloning, sequencing	0		Head (2000)
hybridization, ssDNA nuclease	PCR, array hybridization	0	<10%	Smith *et al.* (1998)
hybridization, primer extension	array	0	<25%	Lipton (1995)
hairpin formation, selective PCR, length selection	PCR, cloning, sequencing	83.8%		
hybridization, ssDNA nuclease	enzymatic cleavage, FRET, array	0	<4%	Smith *et al.* (1998)
subsequence selection (gel capture)	PCR, electrophoresis [c]	0	<20% [c]	Lipton (1995); Roweis *et al.* (1998)
restriction endonuclease digestion	digestion, gel electrophoresis	0		Head (2000)
gel electrophoresis	cloning, gel electrophoresis	4%		Head (2000)
restriction endonucl. dig., PCR, electrophoresis	cloning, sequencing	0		
(gradient) PCR, subseq. sel., gradient electrophoresis	cloning, sequencing	0		Adleman (1994)
hybridization	array	0	<84%	
hybridization, mismatch endonuclease, gel migration	electrophoresis, FCS	17% (gel) 0 (FCS)	<69% (enz.) <26% (FCS)	Rozenberg & Spaink (2003)
translation	MALDI-TOF mass spectrometry	0		Head (2000)
gel electrophoresis, translation	cloning, electrophoresis	20–40%		Head (2000)

b Error rate: the number or percentage of incorrect answers recovered.
Readout ambiguity: maximum ratio of signal of incorrect answer to signal of correct solution

c Braich *et al.* (2001) used sequencing as the output strategy, and determined a readout ambiguity of $6 \times 10^{-5}\%$

An interesting fact about the computations listed is that nearly all problems have been designed to produce just one unique solution (Adleman, 1994; Ouyang *et al.*, 1997; Aoi *et al.*, 1998; Lee *et al.*, 1999; Yoshida & Suyama, 2000; Head *et al.*, 2000, 2002a; Sakamoto *et al.*, 2000; Braich *et al.*, 2001, 2002; Wang *et al.*, 2001; Lee *et al.*, 2003, 2004; Takenaka & Hashimoto, 2003). While this does demonstrate the power of the selection and detection technology, it is not necessarily a realistic scenario for many applications, including solutions to hard problems. The multiple solutions possible in the reports by Pirrung *et al.* (2000), Liu *et al.* (2000), Faulhammer *et al.* (2000) and Head *et al.* (2002b) require an extra step in the computation to physically separate these solution molecules. Obvious approaches are transformation to bacteria (Ouyang *et al.*, 1997) and hybridization to an addressed array (Liu *et al.*, 2000), while dilution prior to PCR may also work (Braich *et al.*, 2001).

Several implementations use immobilized DNA (Adleman, 1994; Smith *et al.*, 1998; Morimoto *et al.*, 1999). While this moving to two dimensions (DNA arrays) or '2½' dimensions (beads) theoretically limits the parallelism that can be achieved, it does provide additional control over molecules, for example of their position and reactivity.

Several facts listed in table 1 deserve some further attention. First, these data should be regarded primarily as an illustration of the diversity of molecular algorithms, and less as a basis for complexity comparisons between them. Not every step in every algorithm has a counterpart in others, and the distinction between steps may not be equally accurate or relevant for every implementation. This is particularly evident with the implementation of Yoshida & Suyama (2000), where selection already occurs during every step of the generation phase. These smart heuristics considerably reduce the number of complete solutions that need to be evaluated, whereas all other approaches still use brute force methods.

Second, the implementations of Liu *et al.* (2000) and Wang *et al.* (2001) appear to be somewhat anomalous, as the problem instances are not actually solved by molecules but rather by the computer operators. Because the potential solutions used have no intrinsic bit representation, a lookup table must be consulted at every operation, and the operator must determine which solutions are satisfying. These 'prototype' implementations are therefore primarily demonstrations of molecular control. Wang *et al.* (2000) propose a 'multiple-word' extension of the architecture (with experimental evidence for a two variable SAT problem), which solves this issue and increases the scaling potential of the architecture. Su & Smith (2004) provide further corroboration for the versatility of the architecture by implementing logic gates on a surface.

Alternative modes of DNA based computation

Apart from optimization problems, DNA computing has also been applied to a variety of other problems. The techniques used are similar, and so are the challenges: to control the reactions and reliably detect correct output molecules. Two experimental DNA databases have been presented, both of considerable size: 1.7×10^7 (Brenner *et al.*, 2000) and 3.6×10^7 (Reif *et al.*, 2002). The former was even loaded with biologically relevant cDNA, coupled to a DNA address label. In both cases, the library strands are attached to microscopic beads for easy handling and synthesis. Queries upon the databases are implemented by adding fluorescently labelled complementary DNA. The whole library is then sorted by flow cytometry, and the most fluorescent (approximately 1%) fraction is recovered. The filtering is therefore more crude than the methods described in the previous section, however this fuzziness also enables recovery of strands very much like the one asked for, which may be interesting for some applications.

Another branch of experimental DNA computing is concerned with implementation of finite automata, a simple model of computation. This model consists of a machine, which can assume a finite number of states, and transition rules. The rules specify the next state based on current state and input, and the computation terminates when the machine reaches a specified final state. Finite automata can only perform relatively simple computations, in contrast to a Turing machine, which is more complicated in structure but capable of universal computation. The molecular computations are typically autonomous, i.e. components corresponding to the initial state and rules are mixed, and the computation proceeds to the output molecule through a cascade of reactions. Several different designs have been tested experimentally, one based on restriction enzymes and ligase (Benenson *et al.*, 2001; Benenson *et al.*, 2003) and similar to a design for a molecular Turing machine by Rothemund (1995). Another design uses 'whiplash' PCR, in which state transitions are encoded on a single molecule and executed by intramolecular priming during PCR (hairpin formation; Sakamoto *et al.*, 1999; Komiya *et al.*, 2001). Third, rationally designed deoxyribozymes (Breaker, 2000) have been used to construct an automaton that was capable of playing tic-tac-toe perfectly (Stojanovic & Stefanovic, 2003). Finally, assembly of DNA binding proteins on single-stranded DNA was interpreted as a finite state machine (Bar-Ziv *et al.*, 2002). All these designs are based on the chance encounters between large numbers of molecules in solution (Brownian search), but do not exploit the parallelism inherent in the numbers of different molecules possible. A combination of the two approaches, parallel search and automata, could yield more powerful DNA computers: automaton logic might be used to process complex mixtures of DNA. This is close to the envisaged biomedical or biosensory applications of molecular automata (Stojanovic & Stefanovic, 2003; Benenson *et al.*, 2004).

A completely different type of biomolecular computing device is based on genetic regulatory networks inside cells (Simpson *et al.*, 2001; Weiss *et al.*, 2002; Hasty *et al.*, 2002). In this approach, genetic control elements are employed in the construction of *in vivo* logic gates, which can be further integrated into genetic circuits. Typically, concentrations of gene products are taken as signals: above a certain concentration threshold, a signal is interpreted as '1', below as '0'. Such systems based on reaction kinetics are in theory sufficient for the construction of universal computers and neural networks (Hjelmfelt & Ross, 1995). The motivations for this type of genetic engineering are close to those of molecular automata: artificial logic circuits can be employed to wire cells as biosensors, and as components in logical gene therapy. Basic logic gates have already been constructed (Gardner *et al.*, 2000; Yokobayashi *et al.*, 2002; Hengen *et al.*, 2003; Weiss *et al.*, 2003). The main challenge is to get these to work together reliably in circuits.

There has long been an interest in DNA as a structural material for nanotechnology (Seeman, 1999, 2003). DNA computing principles are very promising in the construction of supramolecular nucleic acid structures, as they allow for programmable interactions between building blocks. The first demonstration of this philosophy was the self-assembly of two-dimensional lattices using cross-over DNA 'tiles' (Winfree *et al.*, 1998). Such tiles are rigid constructions consisting of several intertwined helices, and with four sticky ends ('sides') that can be used to guide their assembly. Tiles in general can be used to model computations, with some systems equivalent in power to a Turing machine. This relation has been exploited to perform a computation (four step cumulative logical XOR) using DNA tiles (Mao *et al.*, 2000).

Most supramolecular structures created so far are crystalline in nature, i.e. there is a simple periodicity of elements (Winfree *et al.*, 1998; Seeman, 2003; Chworos *et al.*, 2004). To be useful in nanoscale engineering, more structure must be programmed into the assembly. Recently, two studies have shown how the aperiodicity of the DNA nucleotide sequence can be translated to supramolecular aperiodic crystals – more specifically, small barcode assemblies (Yan *et al.*, 2003) and fractal triangles (Rothemund *et al.*, 2004). Another application of DNA structural engineering are DNA based molecular nanomachines (Yurke *et al.*, 2000; Yan *et al.*, 2002; Niemeyer & Adler, 2002), the design of which also shares principles with DNA computers.

Finally, some attention has been given to explicitly performing arithmetic using DNA (Guarnieri *et al.*, 1996; Oliver, 1997; Yurke *et al.*, 1999; Hug & Schuler, 2002). Carrying out calculations is central to silicon based computers, and in comparison DNA calculates poorly, with experimental evidence for molecular arithmetic limited to 1+1 (=2; Guarnieri *et al.*, 1996) and 3+3 (=6; Yurke *et al.*, 1999). However, such calculations may prove useful precisely because they pro-

vide extra control capabilities for molecular scale actions. For example, DNA nanotechnological efforts may benefit from the capability to deposit precise numbers of molecules.

Implementation issues

A number of difficulties arise in the implementation of computations in DNA. Many computations suffer critical errors because of undesired molecular interactions. There are several reasons for this, the most important being intrinsic molecular behaviours and suboptimal protocols. Most of molecular biology is not concerned at all with 100% reliability, as most computing purposes require: DNA handling protocols are designed for acceptable results in reasonable time, where acceptable may even signify a 5% success rate for some operations. DNA computing is therefore a catalyst in the optimization of protocols for reliability, reproducibility and accuracy (see also table 2). However, handling molecules is still largely a stochastic activity, and anomalous behaviour is likely to be unavoidable.

An important issue in the implementation of DNA computers is careful sequence design. The basic programming of the computer is often held in the nucleotide sequence, but other considerations also affect sequence choice. Depending on the design of the DNA computer, different types of molecular interaction are required which put constraints on sequence design (Brenneman & Condon, 2002; Dirks *et al.*, 2004; Mauri & Ferretti, 2004). Examples are melting behaviour, subsequence distinctiveness, enzyme recognition sites and three-dimensional folding. Several software packages have been developed to aid in sequence design (Feldkamp *et al.*, 2003; Kim *et al.*, 2003), and even a DNA computation has been performed to design suitable DNA computing sequences (Deaton *et al.*, 2003; Chen *et al.*, 2004).

Computations such as those listed in table 1 are still performed by manual pipetting. This hinders scaling of computations to larger instances, but it also introduces human error. Several strategies for the automation of computations have been presented, e.g. liquid handling robotics (Hagiya, 2001) and microfluidic lab-on-a-chip procedures (Gehani & Reif, 1999; McCaskill, 2001; Chiu *et al.*, 2001). Application of such technology will certainly enhance the power and reliability of DNA computing. Another strategy to avoid human error is to let the computations proceed autonomously, without the need for intervention between input and output. The computations that can be performed this way are currently limited to those dependent on state transitions (Benenson *et al.*, 2001) and self-assembly (Mao *et al.*, 2000).

A last important drawback of current DNA computation concerns the output mechanisms. Output molecules are generally analysed using crude detection methods (predominantly gel electrophoresis, see table 2). Sensitive high-throughput screening technologies need to be developed in order to overcome the limitations imposed by existing readout methods.

Evolutionary algorithms for DNA computers

In summary, to apply DNA computing to large combinatorial optimization problems is promising in theory, but quite a challenge to implement in practice. Aside from technical difficulties, fundamental restrictions on such approaches are biochemical errors, search spaces required and lack of speed. It has been proposed (early on by Stemmer, 1995, and Adleman, 1996, later in more detail by Deaton *et al.*, 1997; Chen & Wood, 2000; Bäck *et al.*, 2003) that these limitations may be circumvented by the implementation of evolutionary algorithms in DNA computers. Using careful encoding, such systems could take advantage of biochemical noise by using it as a source of variation. Iteration of a selection cycle could yield a directed search through sequence space, foregoing the task of checking every possible solution. Evolutionary DNA computing could be modelled after *in vitro* directed evolution, but with different selection criteria (figure 5).

Evolutionary DNA computers are potentially much more powerful than *in silico* evolutionary problem solving approaches. Advantages may include vastly larger population sizes (10^{12} *in vitro*, typically 10^{3} *in silico*), better evolutionary performance (*in silico* evolutionary algorithms are abstractions of biochemical realities) and true non-determinism.

Directed molecular evolution

In vitro evolution has proved to be a valuable strategy in searching for macromolecules with certain desired properties. The approach uses recursive cycles of recombination, mutation and selection on a population of candidate nucleic acids to enrich this pool in 'successful' species. Searches can be conducted for novel or improved protein functions (Minshull & Stemmer, 1999), as well as for (deoxy)ribozymes and aptamers (nucleic acid ligands; Wilson & Szostak, 1999; Joyce, 2004). The search spaces for such methods are of astronomical proportions, making *de novo* design of functional molecules all but impossible. In fact, models of protein design are known to be NP-complete problems (Fraenkel, 1999; Pierce & Winfree, 2002). Recursive selection and recombination can 'compute' answers to such molecular design problems (Stemmer, 1995), suggesting that other hard problems may well be open to molecular evolutionary optimization.

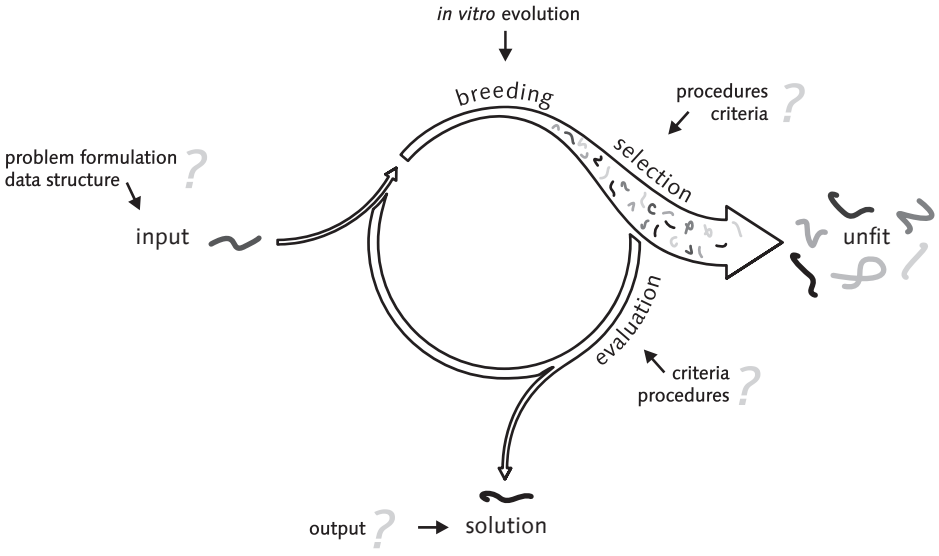

Figure 5. Evolutionary DNA computing. Candidate solutions are generated by amplification and diversification through mutation and recombination (molecular breeding). A selection procedure filters out the 'fittest' candidates. If these are satisfactory, the computation ends; if not, they are used as input for another iteration of the cycle. Implementation issues are indicated by arrows.

Apart from providing theoretical support, *in vitro* evolution may be of practical relevance for evolutionary DNA computing. Especially protocols for the generation of molecular diversity, by for instance artificial recombination (DNA shuffling; Stemmer, 1994), should prove useful.

Outline of this thesis

The aim of the research reported in this thesis was to explore the feasibility of practical evolutionary DNA computing. So far, no implementations or feasible designs have been reported. The only published experimental results concern a technique (two-dimensional gradient gel electrophoresis) that has been proposed as a selection procedure (Wood *et al.*, 1999; Goode *et al.*, 2001).

It is not known which algorithms, methods and selection criteria might prove useful, or which types of problems could be solved by evolutionary DNA computations (see also figure 5). Probably the only well investigated module is the breeding phase, for which amplification, recombination and mutation methods can be collected from directed molecular evolution experience.

The computations listed in tables 1 and 2 serve as an example of the difficulties involved. Most implementations require iterated selection procedures for local

properties of the solution molecules, with the number of iterations dependent on the problem size. These complicated selections are equivalent to the combined selection and evaluation in a single evolutionary cycle (figure 5). As the computations listed represent the state of the art in selection, it is clear that at present repeated cycles are not feasible. Selection procedures for evolutionary DNA computations should consist of a limited number of steps (ideally only one), but they may allow for larger errors than those used in deterministic computation. The evaluation procedure should be equally limited in time, but needs to be more precise.

Another perspective on the problem considers the data structures and test problems used. Again, current DNA computations offer few openings. The majority relies on either a surface-based or Lipton architecture (figure 4), both of which appear currently inadequate as candidates for evolutionary DNA computing. The surface-based methods suffer from an intrinsic bound on evolutionary search space: the number of iterations of the evolutionary loop is determined by the chosen surface area instead of by the appearance of satisfactory solutions. Still, if methods are developed to generate and recombine strands on a surface, immobilized DNA might support evolutionary searches. The Lipton encoding would be more difficult to adapt, as it is fundamentally dependent on local properties (subsequences) of the solution strands. Populations would have to be subjected to serial subsequence inspection for every selection iteration, which is an unfeasible scenario.

The research reported in this thesis consists of several DNA computations to optimization problem instances, using a variety of experimental techniques, algorithms and selection criteria. Some of these may prove of use in the eventual implementation of evolutionary algorithms. For completeness, these computations are included in tables 1 and 2.

Chapter 2 explores the use of several techniques for the detection of DNA hybridization, which may be a good selection criterion (phenotype) for evolutionary DNA computing (Wood *et al.*, 1999; Goode *et al.*, 2001). Hybridization detection methods are routinely used for other applications, but it is uncertain whether they are reliable enough for computing purposes. Heteroduplex migration and mismatch endonuclease assays were adapted from mutation detection protocols and tested on a small instance of 3SAT. Fluorescence energy resonance transfer, a technique that can be applied to study molecular interactions, was also tested on several DNA combinations.

The experiments in chapter 3 extend this use of fluorescent labelling for hybridization detection. Fluorescence cross-correlation spectroscopy, an extremely sensitive detection method, was employed to monitor a DNA computation at the level of single molecules. The experiments in chapters 2 and 3 represent the first laboratory implementations of a new algorithm for molecular computing, which

has the advantage of requiring only a single selection step on global properties of the solution molecules (Rozenberg & Spaink, 2003).

Another detection technique, mass spectrometry, is applied to detect the outcome of the computation in chapter 4. However, a protein representation instead of DNA is analysed. As in natural systems, proteins may provide a good phenotype for an evolutionary search. Chapter 5 also uses this translation principle, in conjunction with a very straightforward selection criterion, DNA length. It is shown how the two may be combined to enable multi-criterion optimization.

Chapter 6 summarizes the results, evaluates the methods for use in evolutionary DNA algorithms, and discusses the future prospects of DNA computing.

2

Molecular implementation of the blocking algorithm

Based on:

C.V. Henkel, G. Rozenberg & H.P. Spaink (2004) Application of mismatch detection methods in DNA computing. In: C. Ferretti, G. Mauri & C. Zandron (eds.) *Tenth international meeting on DNA computing, preliminary proceedings*. Università di Milano-Bicocca, pp 183–192

K.A. Schmidt, C.V. Henkel, G. Rozenberg & H.P. Spaink (2002) Experimental aspects of DNA computing by blocking: use of fluorescence techniques for detection. In: R. Kraayenhof, A.J.W.G. Visser & H.C. Gerritsen (eds.) *Fluorescence spectroscopy, imaging and probes – new tools in chemical, physical and life science*. Springer-Verlag, Berlin Heidelberg, pp 123–128

Abstract

In many implementations of DNA computing, reliable detection of hybridization is of prime importance. We have applied a fluorescence technique and several well-established mutation scanning methods to this problem. All these technologies are appealing for DNA computing, as they have been developed for both speed and accuracy. Fluorescence resonance energy transfer was tested as a hybridization detection method on several combinations of oligonucleotides. A heteroduplex migration assay and enzymatic detection of mismatches were tested on a four variable instance of the 3SAT problem, using a previously described blocking algorithm. The heteroduplex method is promising, but yielded ambiguous results. On the other hand, we were able to distinguish all perfect from imperfect duplexes by means of a CEL I mismatch endonuclease assay.

Introduction

Computing by blocking is a recently described methodology for molecular computing (Rozenberg & Spaink, 2003). The blocking algorithm uses nucleic acid complementarity to remove molecules not representing a solution from the candidate pool. To an initial library of single-stranded DNA molecules corresponding to (all) potential solutions, a set of complementary falsifying DNA (blockers) is added. Only those library molecules not representing solutions will combine with a blocker to form a perfect DNA duplex. Library molecules corresponding to solutions should remain single-stranded or form a duplex with mismatched basepairs, depending on experimental conditions. The experimental challenge in implementing this algorithm is to very precisely separate perfectly matched molecules from mismatched ones.

The original proposal for the implementation of the blocking algorithm was using PCR inhibition. Molecules not satisfying the 3SAT instance were to be made unavailable for DNA polymerase through their association with a blocker molecule, for example peptide nucleic acid (PNA). This would result in the selective amplification of unblocked DNA. So far, experimental data supporting this method is lacking.

Here, we report the use of fluorescence resonance energy transfer (FRET), a heteroduplex migration assay and enzymatic mismatch recognition to implement blocking. The former has already been used in combination with oligonucleotides (Cardullo *et al.*, 1988). The latter two techniques are widely used to scan for mutations in molecular biological and clinical laboratories, and are well suited for high-throughput analysis of large numbers of samples (Taylor, 1999; Kristensen *et al.*, 2001).

A fluorophore in the excited state can transfer its excitation energy non-radiatively to another unexcited fluorophore, if it is in very close proximity. The net result is quenching of the emission of the first (donor) fluorophore, and appearance of emission from the second (acceptor). The efficiency of the FRET phenomenon is highly dependent on the distance between the two molecules. The efficiency of energy transfer E is given by

$$E = \frac{R_0^6}{R_0^6 + r^6},$$

where r is the distance, and R_0 is the Förster distance (Lakowicz, 1999). R_0 is dependent on the fluorescence characteristics of the specific dye couple used, and is defined as the distance at which energy transfer is 50% efficient (typical values are of the order of 50 Å). Because of the high dependence on distance, FRET can be used as a molecular ruler, and to study interactions between molecules. If two molecules are fluorescently labelled, FRET will only occur if the fluorophores are brought close together, i.e. by binding between the molecules. Unbound molecules in solution are too far distant to engage in detectable energy transfer. DNA hybridization is also capable of bringing donor and acceptor in FRET range (Cardullo *et al.*, 1988), a concept that has been exploited for the design of novel DNA probes, for instance molecular beacons (Tyagi *et al.*, 1998).

Hybridization detection by FRET relies solely on hybridization kinetics. In contrast, heteroduplex migration and enzymatic cleavage are dependent on DNA spatial structure. During electrophoresis, perfect double-stranded (homoduplex) DNA migrates through a gel at a predictable rate, dependent only on the strength of the applied electrical field, gel and buffer conditions and DNA length. However, DNA containing nucleotide mismatches (heteroduplex) and single-stranded DNA migrate at anomalous rates, caused by secondary structure formation (ssDNA) or helix distortion (dsDNA). Such structures experience specific, but unpredictable, resistances when migrating through the gel matrix. Heteroduplex mobility is lower than that of homoduplexes of equal length and as a result bands end up higher on the gel; single strands migrate faster. Several well-established and sensitive mutation detection techniques exploit this effect, such as single strand conformational polymorphism (SSCP), temperature or denaturing gradient gel electrophoresis (TGGE, DGGE) and heteroduplex analysis (Nataraj *et al.*, 1999).

Enzymatic mismatch recognition is also widely used in mutation detection (Mashal *et al.*, 1995). It uses specific endonucleases which recognize and digest the abnormal DNA conformations which result from mismatched nucleotides. We have used the recently discovered CEL I nuclease, purified from celery, for this purpose (Oleykowski *et al.*, 1998; Yang *et al.*, 2000).

Working principles of the three methods are illustrated in figure 1.

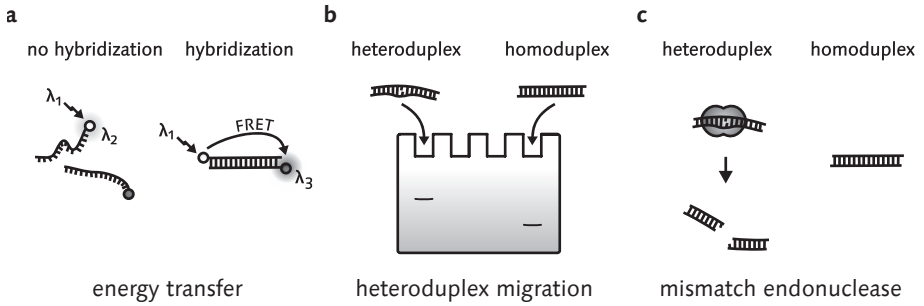

Figure 1. Principles of hybridization detection. **a** Fluorescence energy resonance transfer. If two fluorophores with overlapping spectra are brought into close proximity, for example through DNA hybridization, excitation energy can be transferred from one fluorophore to the other. The net result is a shift in emission wavelength ($\lambda_2 \rightarrow \lambda_3$). **b** Gel migration. Heteroduplexes have an anomalous helix structure, and therefore migrate slower through a gel than homoduplexes. **c** Mismatch endonucleases. Only DNA containing mismatched nucleotides is digested by the enzyme.

Materials and methods

Sequence design

To represent the entire solution space to a four variable SAT problem, 16 library oligonucleotides were designed. The general structure of the library molecules is:

$$5' \; [start][a][b][c][d][stop],$$

with *a*, *b*, *c* and *d* variable sequences representing variables. Two subsequences correspond to the two values these variables can take. The sequence of any variable thus only depends on its value, not on its identity. *start* and *stop* are invariable sequences. Library molecules are numbered from 0 to 15, after the binary numbers they encode. For example, truth assignment *abcd* = {1010} is represented by oligonucleotide 10. Falsifying oligonucleotides, or blockers, are complementary to the library oligonucleotides:

$$3' \; [start][a][b][c][d][stop].$$

Since the falsification of a clause only requires three specified variables, and blocker molecules must contain a statement on all four variables, two blockers need to be designed for every clause. The fourth variable is set to true in one, and to false in the other. (It may be possible to circumvent this encoding com-

plication through the use of redundant blockers, which contain universal nucleotides; Loakes, 2001.) The translation of all clauses into blockers is summarized in table 1.

Different value subsequences were used for the experiments described here. For the FRET experiments, these are CTT for *false*, and CAT for *true*. *start* and *stop* are single nucleotides, T and C, respectively. Only two library molecules were tested: 04 (T CTT CAT CTT CTT C) and 07 (T CTT CAT CAT CAT C), representing truth assignments *abcd* = {0100} and {0111}, respectively. Two blocker molecules were tested, A0 (falsifying *abcd* = {1010}, sequence 5' G AAG ATG AAG ATG A) and B0 (falsifying *abcd* = {0100}, sequence 5' G AAG AAG ATG AAG A).

Value sequences were primarily selected for isothermal melting characteristics, i.e. the melting temperature (T_m, the temperature at which 50% of the DNA exists as dsDNA) of every perfect duplex is identical, irrespective of its computational value. Melting temperatures were calculated according to SantaLucia (1998) and Peyret *et al.* (1999). Furthermore, sequences should be as short as possible to enable energy transfer between both 5' fluorophores. In a first experiment, 3' labelling was used for the blocker molecules and 5' for the library. This approach brings the dyes in close proximity, independent on DNA length. However, this resulted in strong quenching of both fluorophores, most likely due to exciton interaction (Bernacchi & Mély, 2001; Bernacchi *et al.*, 2003). As a final constraint, the number of guanine residues should be kept low to avoid quenching of some dyes (Seidel *et al.*, 1996; Nazarenko *et al.*, 2002). Value sequences were chosen after exhaustive evaluation of all two and three basepair possibilities.

For the gel migration and enzymatic cleavage assays, as well as the single molecule experiments described in chapter 3, other value sequences were used: ATCACC for *false*, and GTCTGA for *true*. *start* and *stop* sequences (CTTGCA and TTGCAC, respectively), bring the total length of the molecules to 36 nucleotides. Complementary blocker sequences are *start* = GAACGA, *stop* = AACGTG, *true* = CAGACT and *false* = TAGTGG (all 3'→5'). Sequences used are listed in tables 1 and 2 (results section).

Constraints on the design of the variable sequences were slightly different from those described above for the FRET molecules. Here, length of oligonucleotides was not limited by a Förster distance, so slightly longer subsequences were chosen. This allows for a higher mismatch ratio in non-complementary molecules: variable sequences vary by four out of six nucleotides, instead of one out of three as before. Invariable sequences were also elongated to avoid 'frameshifting' (for example, hybridization of library sequence *a* to blocker sequence *b*). Other constraints were: GC content <50 %, isothermal melting behaviour, no repeats or subsequence complementarity >2 bp, and no self complementarity.

Oligonucleotides

Oligonucleotides were custom synthesized and labelled at Isogen Bioscience (Maarssen, The Netherlands) and Eurogentec (Seraing, Belgium). Molecules for FRET measurements were 5' labelled, library molecules by fluorescein (isothiocyanate derivative, Molecular Probes) and blockers by TAMRA (tetramethylrhodamine, Molecular Probes). Concentrations were calculated from absorption measurements of the dyes at 494 nm (fluorescein) or 555 nm (TAMRA), assuming molar extinction coefficients of 77,000 cm^{-1} M^{-1} (fluorescein) and 83,000 cm^{-1} M^{-1} (TAMRA). These oligonucleotides were used without further purification.

Library molecules for gel migration and enzymatic cleavage assays contain a covalent 5' Cy5 label (Amersham Biosciences), blockers a 5' fluorescein (FITC, Molecular Probes). All oligos were purified from 10% denaturing polyacrylamide gels to remove unbound dye. DNA was allowed to diffuse from gel slices by overnight soaking in 0.5 M NH_4Ac, 2 mM EDTA, 0.1% SDS, and recovered by ethanol precipitation. Concentrations were calculated from absorption measurements of the dyes at 494 nm (fluorescein) or 649 nm (Cy5). Molar extinction coefficients of 77,000 cm^{-1} M^{-1} (fluorescein) and 250,000 cm^{-1} M^{-1} (Cy5) were used.

Fluorescence measurements

Fluorescence spectra were recorded using a Perkin Elmer LS50B Luminescence Spectrometer. Temperature was regulated by a circulating water bath. Measurements were made in 1× SSC buffer (150 mM NaCl, 15 mM sodium citrate, pH 7.0). Samples were heated to 95 °C for five minutes and cooled on ice prior to measurements. Oligonucleotide concentrations were 1.4 µM for library molecules (04 and 07), 1.8 µM for blocker A0 and 1.6 µM for B0. Other ratios produced similar effects (tested with 2.4 and 3.2 µM B0).

Duplex migration assay

Mixtures of library and blocker molecules were made by combining 5 pmol per oligo in a gel loading buffer consisting of 1× TBE (90 mM Tris-borate, 2 mM EDTA, pH 8.3), 3.3% sucrose and 0.033% Orange G. Duplex DNA was formed by heating the mixtures to 95 °C for 5 minutes, and cooling to 4 °C at 0.1 °C $second^{-1}$ in a thermocycler (Biometra TGradient). Gels were prepared from regular acrylamide:bisacrylamide (20:1) or proprietary SequaGel MD (Mutation Detection) acrylamide matrix (National Diagnostics, Atlanta, Georgia, USA). Duplex destabilizing chemicals (urea, ethylene glycol, formamide, or glycerol) were sometimes added to enhance heteroduplex migration effects (Ganguly *et al.*, 1993). Gels were run in 1× TBE at 200 V and 4 °C. Gel images were captures on a Biorad

FluorS MultiImager, using UV excitation with 530 nm band pass and 610 nm long pass filters for detection of fluorescein and Cy5 fluorescence, respectively. Contrast levels of digital images were adjusted in Corel Photopaint 11.

Enzymatic mismatch cleavage assay

Duplexes were prepared as described above, except that hybridization was carried out in 10 mM Tris/HCl pH 8.5. T7 endonuclease I (T7EI) was obtained from New England Biolabs and handled according to the manufacturers recommendations. Reactions containing 5 pmol per oligonucleotide and 1 unit of enzyme were allowed to proceed for up to 150 minutes.

CEL I enzyme was obtained from Dr Edwin Cuppen (Hubrecht Laboratory, Utrecht, The Netherlands), see *http://cuppen.niob.knaw.nl* for a detailed isolation protocol. Several batches of varying activity were used throughout the experiments described in this chapter. Every lot of CEL I was tested, and for all subsequent experiments quantities were used that gave the effect shown in figure 3 after 30 minutes of incubation. Reactions were performed with 5 pmol per oligonucleotide in a 4 µl volume at 45 °C, in a 10 mM $MgSO_4$, 10 mM HEPES pH 7.5, 10 mM KCl, 0.002% Triton X-100, 0.2 µg $µl^{-1}$ BSA buffer. Reactions were stopped by placing samples on ice and adding 4 µl 80% formamide, 100 mM EDTA. Digests were analysed on 10% TBE/polyacrylamide gels, which were imaged as before. Bands were analysed using ImageJ software (version 1.31v, *http://rsb.info.nih.gov/ij*).

Table 1. Blocker molecules

Clause	Falsified by abcd	Blocker molecule	Sequence (5'→3') [a]
$\neg a \vee b \vee \neg c$	1010	A0	GTGCAA GGTGAT TCAGAC GGTGAT TCAGAC AGCAAG
	1011	A1	GTGCAA TCAGAC TCAGAC GGTGAT TCAGAC AGCAAG
$a \vee \neg b \vee d$	0100	B0	GTGCAA GGTGAT GGTGAT TCAGAC GGTGAT AGCAAG
	0110	B1	GTGCAA GGTGAT TCAGAC TCAGAC GGTGAT AGCAAG
$\neg a \vee c \vee \neg d$	1001	C0	GTGCAA TCAGAC GGTGAT GGTGAT TCAGAC AGCAAG
	1101	C1	GTGCAA TCAGAC GGTGAT TCAGAC TCAGAC AGCAAG
$b \vee c \vee \neg d$	0001	D0	GTGCAA TCAGAC GGTGAT GGTGAT GGTGAT AGCAAG
	1001	identical to C0	

a Sequences are for the gel migration and enzymatic cleavage experiments only

Results

Problem instance and algorithm

We have tested mutation detection techniques on the following four variable, four clause 3SAT satisfiability problem:

$$F = (\neg a \vee b \vee \neg c)\ \&\ (a \vee \neg b \vee d)\ \&\ (\neg a \vee c \vee \neg d)\ \&\ (b \vee c \vee\ \neg d),$$

where *a*, *b*, *c* and *d* are the four variables with values of *true* (1) or *false* (0), $\vee$ stands for the *or* operation, & for *and*, $\neg$ for negation. Since the clauses are connected by *and*, falsifying one clause is sufficient for falsification of the entire formula. For example, falsification of the first clause by *abc* = {101} falsifies the complete formula *F*.

The blocking algorithm proceeds as follows:

1. synthesize all possible assignments as ssDNA;
2. synthesize blockers representing falsifying assignments;
3. mix and hybridize;
4. apply mismatch detection method.

The library/blocker combinations that form perfect dsDNA correspond to false assignments.

Hybridization detection by FRET

Energy transfer measurements were only performed for the four combinations of libraries 04 and 07 and blockers A0 and B0. Figure 2 shows emission spectra obtained with excitation of fluorescein at 460 nm. At low temperature, all combinations are able to associate. This results in quenching of fluorescein emission around 520 nm, and emission through energy transfer of TAMRA around 580 nm. At elevated temperature (above the T_m of a perfect blocking combination), fluorescein quenching is alleviated and TAMRA emission largely disappears. However, emission at 580 nm remains more or less constant as it falls within the shoulder of the fluorescein peak. Therefore, fluorescein quenching is the best indicator of hybridization.

Of the four combinations tested, only one (04+B0) should be able to form a perfect duplex. However, from figure 2, this is not immediately obvious. Figure 3 shows the maximum fluorescein emission for all combinations over a range of temperatures. From this figure, it is clear that 04+B0 is indeed the most stable combination. 04+A0, however, shows intermediate melting behaviour. This combination should have three mismatches (library **010**0 + blocker **101**0), but may form a shorter, three variable duplex with dangling ends through matching subsequences (**010**0 + 1**010**).

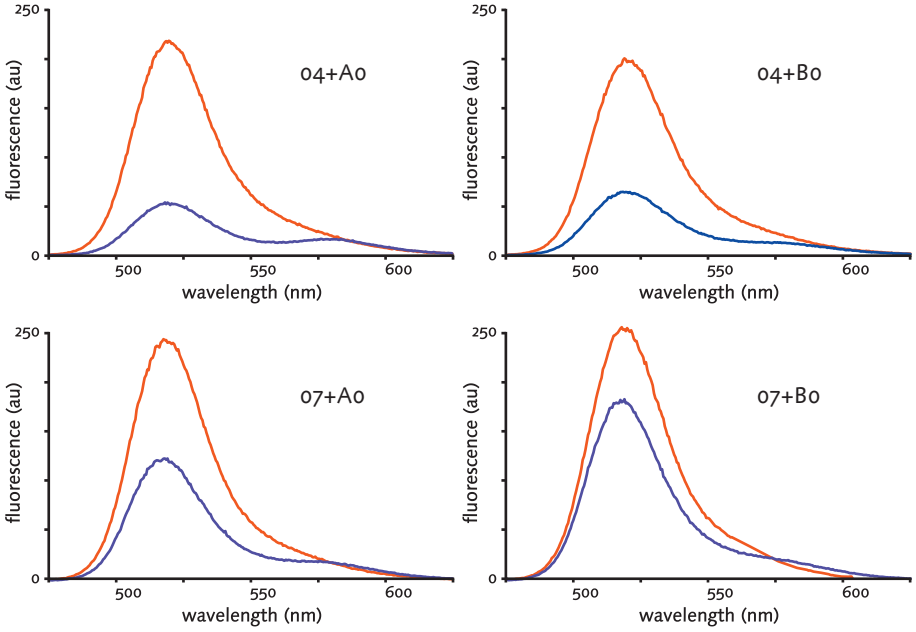

Figure 2. Fluorescence emission spectra of several library/blocker combinations. Spectra were recorded with 460 nm excitation at 22 °C (blue) and 57 °C (red). At high temperature, every sample shows typical fluorescein emission with a peak at 520 nm. At low temperatures, a secondary TAMRA peak is sometimes visible around 580 nm, as well as quenching of fluorescein emission. Fluorescence scale: au, arbitrary units.

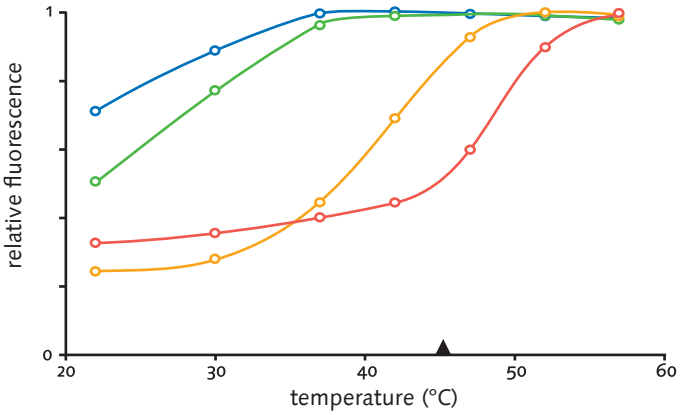

Figure 3. Relative fluorescein emission at different temperatures. Combinations: 04+A0 (orange), 04+B0 (red), 07+A0 (green) and 07+B0 (blue). Fluorescence was measured in the 513–517 nm range, and maximal fluorescence was set to unity for every combination. The arrow indicates the predicted T_m (45.2 °C) for a perfect duplex (SantaLucia, 1998).

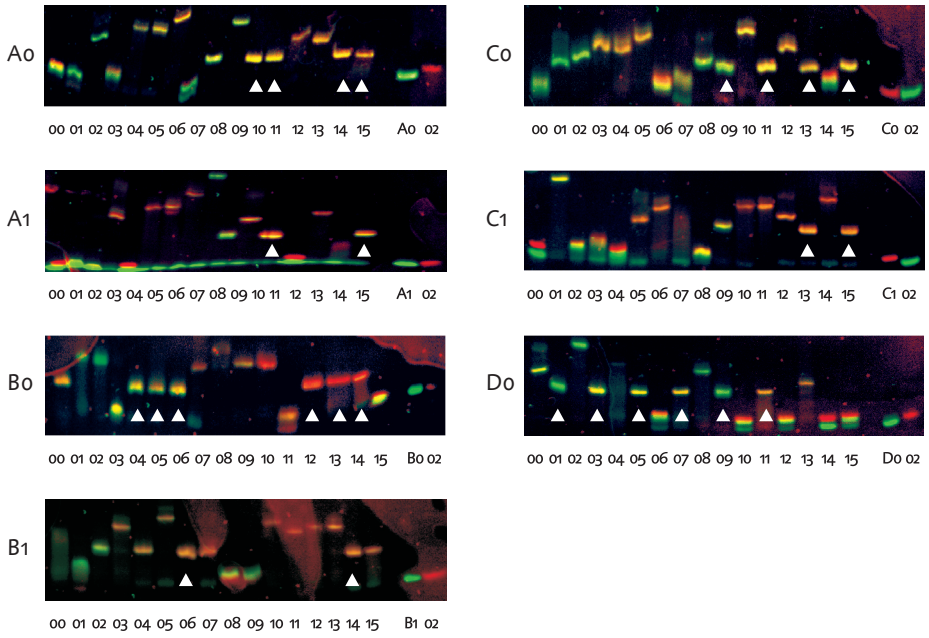

Figure 4. Heteroduplex migration assay for all blocker/library combinations. Each gel contains the complete library (00-15) of oligonucleotides hybridized to the indicated blocker. The rightmost two lanes were loaded with unhybridized blocker and library 02. Images are RGB stacks of the 530 BP (showing the blocker fluorescein label, in green) and 610 LP (library Cy5, red) channels. Duplexes appear as yellow bands, since they fluoresce in both channels at the same location. Red and green bands are non-hybridizing oligonucleotides. Apparent homoduplexes are indicated by arrows.

Heteroduplex migration

Optimal conditions for the heteroduplex migration assay were determined using several blocking and non-blocking oligo combinations and various gel formulations. 12.5% acrylamide gels supplemented with 20% urea were found to give good separation of duplexes and heteroduplexes and were used for all subsequent experiments. Figure 4 shows the gel images for all combinations of blockers with library molecules. Every blocker should only be able to form a perfect duplex with one of the library ligonucleotides, but figure 4 shows up to six apparent homoduplexes per blocker. No improvement was found using MD gel matrix or longer gels (not shown). Nonetheless, some solutions to the satisfiability problem can be identified from figure 4. Library oligos 00, 02 and 08 (*abcd* = {0000}, {0010} and {1000}, respectively) do not behave as a homoduplex in any combination (see table 2).

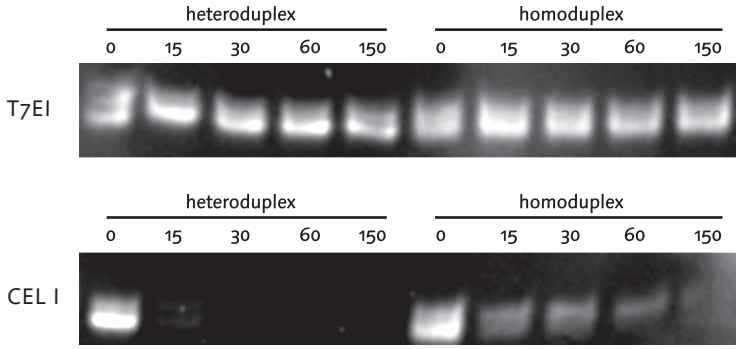

Figure 5. T7EI & CEL I time series. 3 pmol samples of heteroduplex (C0+11) or homoduplex DNA (C0+13) were subjected to both endonucleases for up to 150 minutes. Samples were analysed on a 12% denaturing gel. T7EI does not have any effect on hetero- or homoduplex DNA (here, 0.2 units were used per reaction; 1 unit per reaction gave identical results). CEL I completely degrades the mismatched DNA within 30 minutes. Perfect dsDNA, although also subject to degradation, is still detectable after 150 minutes of reaction.

Mismatch endonucleases

Figure 5 shows the effects of T7EI and CEL I on homoduplex and heteroduplex DNA. Both were incubated for a range of times. In our hands, the T7EI enzyme did not have any discernable effect on any DNA sample, and was therefore not considered for further experiments.

CEL I, however, has a clear effect on all samples. ssDNA is quickly completely degraded (not shown). Homo- and heteroduplexes are both cleaved and broken down, but at different rates. To test whether CEL I would successfully pick blocking from non-blocking combinations, all library molecules were incubated with blockers and enzyme (figure 6). From these results, satisfying assignments could be identified (summarized in table 2).

Figure 6 (facing page). a Effects of CEL I on all blocker/library combinations. Shown are denaturing gels of complete sets of library oligonucleotides and blockers, incubated with CEL I. **b** Quantified fluorescence from the gels. Fluorescence signals from the 530 BP channel are given relative to that of untreated blocker loaded on the same gel. The y-axis shows relative fluorescence, the x-axis the library molecules. Every experiment was executed in duplicate, error bars give standard deviations.

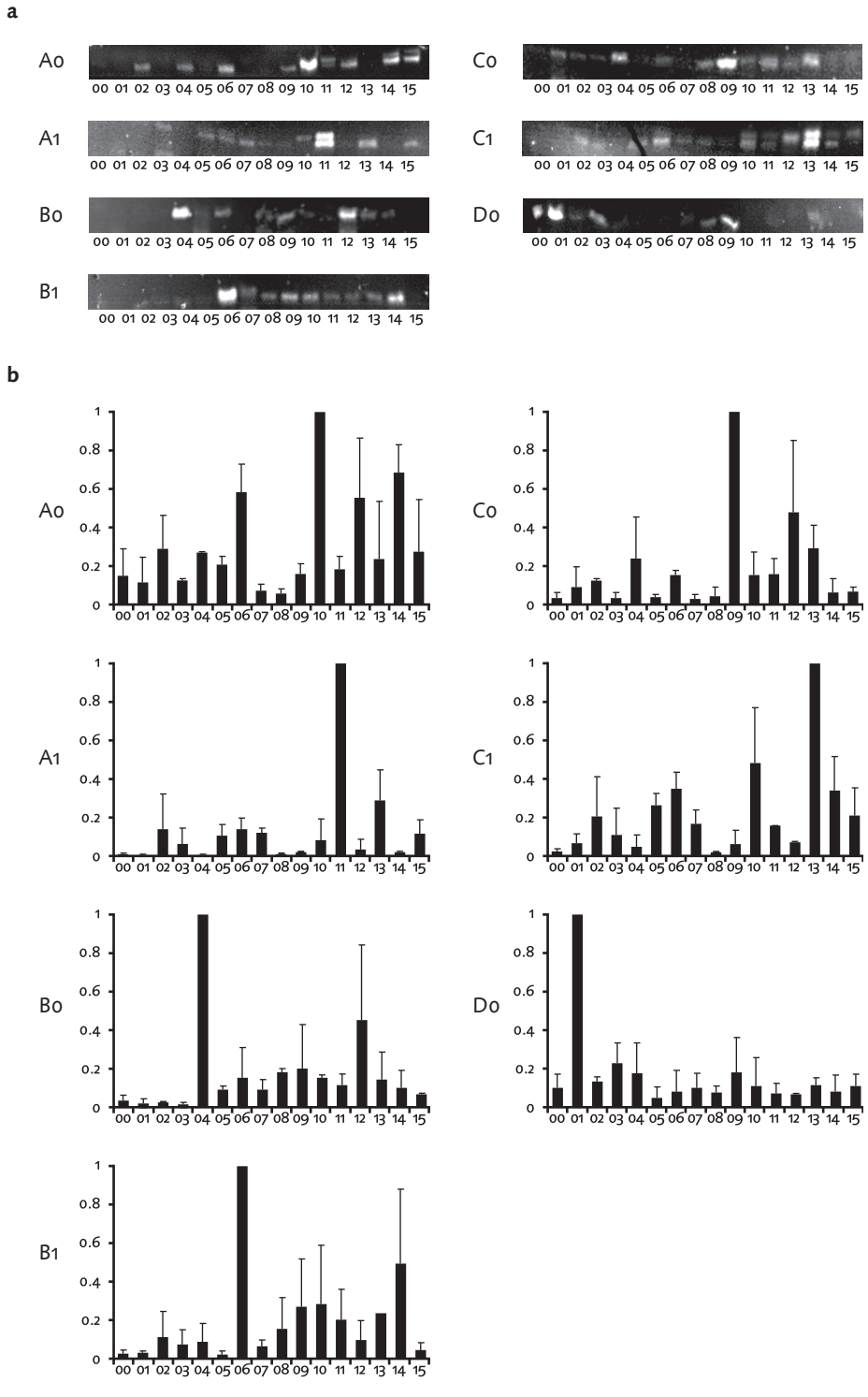
a
Ao
Co
A1
C1
Bo
Do
B1
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
b
Ao
Co
A1
C1
Bo
Do
B1
0
0.2
0.4
0.6
0.8
1
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

Table 2. Apparent solutions to F

Library molecules	abcd	sequence (5'→3')	falsified by: heteroduplex	CEL I
00	0000	CTTCGA ATCACC ATCACC ATCACC ATCACC TTGCAC		
01	0001	CTTCGA ATCACC ATCACC ATCACC GTCTGA TTGCAC	D0	D0
02	0010	CTTCGA ATCACC ATCACC GTCTGA ATCACC TTGCAC		
03	0011	CTTCGA ATCACC ATCACC GTCTGA GTCTGA TTGCAC	D0	
04	0100	CTTCGA ATCACC GTCTGA ATCACC ATCACC TTGCAC	B0	B0
05	0101	CTTCGA ATCACC GTCTGA ATCACC GTCTGA TTGCAC	B0, D0	
06	0110	CTTCGA ATCACC GTCTGA GTCTGA ATCACC TTGCAC	BO, B1	B1
07	0111	CTTCGA ATCACC GTCTGA GTCTGA GTCTGA TTGCAC	DO	
08	1000	CTTCGA GTCTGA ATCACC ATCACC ATCACC TTGCAC		
09	1001	CTTCGA GTCTGA ATCACC ATCACC GTCTGA TTGCAC	C0, D0	C0
10	1010	CTTCGA GTCTGA ATCACC GTCTGA ATCACC TTGCAC	A0	A0
11	1011	CTTCGA GTCTGA ATCACC GTCTGA GTCTGA TTGCAC	A0, A1, C0, D0	A1
12	1100	CTTCGA GTCTGA GTCTGA ATCACC ATCACC TTGCAC	B0	
13	1101	CTTCGA GTCTGA GTCTGA ATCACC GTCTGA TTGCAC	B0, C0, C1	C1
14	1110	CTTCGA GTCTGA GTCTGA GTCTGA ATCACC TTGCAC	A0, B0, B1	
15	1111	CTTCGA GTCTGA GTCTGA GTCTGA GTCTGA TTGCAC	A0, A1, C0, C1	

Discussion

Fluorescence methods are an interesting option for DNA based computation, as they are fast and non-destructive. We have tested one method, based on energy transfer. While in theory capable of detecting hybridization with high sensitivity, the method is limited to rather short oligonucleotides. The system reported here already operates at the limits of FRET possibilities: for the fluorescein/rhodamine donor/acceptor combination, the Förster distance is approximately 50 Å. The dimensions of DNA are 3.4 Å per nucleotide (Seeman, 2002), implying that using 14 basepairs the efficiency of energy transfer is already down to 50% (this is a very straightforward estimate, ignoring the helical turn and fluorophore orientation; see Cardullo *et al.*, 1988, for a more sophisticated model). Since reliable hybridization behaviour is required, these 14 basepairs can encode at most four bits of information. However, as shown by the frameshifting behaviour of combination 04+A0, even longer molecules may be needed, incorporating leader sequences that enforce correct alignment. Yet FRET efficiency decreases dramatically for longer distances (for example, efficiency is down to 1.5% for 100 Å separation).

Solutions for this problem might include other dye combinations (although Förster distances rarely exceed 70 Å; Lakowicz, 1999) and labelling other parts of the oligonucleotides, not just the ends. Still, the latter is unlikely to provide a reliable system for computations as described here, because it would be more sensitive to local hybridization, whereas end labelling allows monitoring global hybridization. Internal labelling introduces the additional difficulty of sequence-specific quenching by nucleotide bases (Nazarenko *et al.*, 2002). Labelling library and blocker oligonucleotides at opposite ends (one 5' and the other 3') will bring the fluorophores in very close proximity upon hybridization, independent of DNA length. This strategy may again accentuate local hybridization, and may facilitate formation of fluorophore heterodimer excitons, with altered spectroscopic features compared to the FRET couple (Bernacchi *et al.*, 2003).

Differential duplex migration also did not provide a suitable test system to distinguish every satisfying solution from non-solutions. There is no general theory describing the effect of anomalous DNA conformations on migration rate, and it is known that not all mismatches can be detected this way (Highsmith *et al.*, 1999; Upchurch *et al.*, 2000). A possible explanation for the ambiguous results reported here is the length of the DNA molecules: heteroduplex migration is generally recommended for DNA 100–500 bp in length. Such lengths also accentuate the effect of a single mismatch, which produces a bend in the helix. In addition, the nature of the mismatches studied here may have contributed. A single variable difference between blocker and library is represented by four non-matching basepairs at a molecular level. These mismatches will probably form a bubble-type configuration, which may not always be subject to higher gel resistances.

We believe that with careful optimization of the encoding, the use of longer molecules (perhaps in combination with scaling to larger problem instances) and more sophisticated analytical techniques (e.g. capillary electrophoresis), the method holds considerable promise. In particular, duplex migration might be employed as a phenotype for the implementation of evolutionary algorithms in DNA (Wood *et al.*, 1999).

The CEL I assay gave more consistent results. However, the results are sometimes difficult to interpret from visual inspection of single gels, because CEL I also degrades perfect duplexes. This breakdown of homoduplex DNA may be due to equilibrium fraying of the molecules, continuously giving the enzyme a toehold on the duplex. Therefore, for this method, longer molecules may also be an option.

Using the blocking algorithm and encoding as reported here, the mismatch endonuclease assay is only useful as an analytical method. Because library molecules that satisfy the problem instance are destroyed, multiple rounds of selection (as in evolutionary algorithms) cannot be easily implemented. However, several other proteins that bind mismatches (such as MutS; Brown *et al.*, 2001) do not destroy the DNA molecule. In future experiments, such proteins may be used in a gel-shift assay (Goode *et al.*, 2001). Besides the enzymatic method tested here, chemical cleavage of mismatches (Bui *et al.*, 2002) could be considered.

Acknowledgements

We thank Dr Edwin Cuppen for the kind gift of purified CEL I enzyme.

3

DNA computing using single-molecule hybridization detection

Based on:

K.A. Schmidt, C.V. Henkel, G. Rozenberg & H.P. Spaink (2004)
DNA computing using single-molecule hybridization detection.
Nucleic Acids Research **32**, 4962–4968

Abstract

Since micromolar DNA solutions can act as billions of parallel nanoprocessors, DNA computers can in theory solve optimization problems that require vast search spaces. However, the actual parallelism currently being achieved is at least a hundred million-fold lower than the number of DNA molecules used. In part, this is due to the quantity of DNA molecules of one species that is required to produce a detectable output to the computations. In order to miniaturize the computation and considerably reduce the amount of DNA needed, we have combined DNA computing with single-molecule detection. Reliable hybridization detection was achieved at the level of single DNA molecules with fluorescence cross-correlation spectroscopy. To illustrate the use of this approach, we implemented a DNA-based computation and solved a four variable, four clause instance of the computationally hard Satisfiability (SAT) problem.

Introduction

For the successful implementation of DNA-based computations, the detection of output molecules is of prime importance. Many of the currently available techniques for detection of DNA have been used in molecular computing: gel electrophoresis with fluorescent or radiometric visualization, fluorescent labelling and fluorescence resonance energy transfer (FRET), mass spectrometry or surface-based techniques. However, all these methods either detect DNA in bulk quantities or destroy the output molecules. This severely limits the size of the library to be searched: the largest parallel computation reported filtered 2^{20} different molecular species (Braich *et al.*, 2002), which is less than the number of molecules of one variety necessary for detection by gel electrophoresis (35 pg; Tuma *et al.*, 1999). This detection limit imposes an equally severe redundancy on any other type of DNA computation. Therefore, the application of a more sensitive detection technology may significantly enhance the power of DNA computations.

Recent progress in optical detectors has enabled the efficient detection of single molecules by fluorescence microscopy (Weiss, 1999). One of the most prominent single-molecule techniques for biological research is fluorescence correlation spectroscopy (FCS; Madge *et al.*, 1972; Eigen & Rigler, 1994). FCS studies fluorescence fluctuations caused by single molecules diffusing in a focal detection volume. Since binding of a small fluorescently labelled molecule to a larger ligand results in a change in diffusion time, FCS allows quantification of the interaction of biological molecules at extremely low concentrations. Extension of the method to dual-colour fluorescence cross-correlation spectroscopy

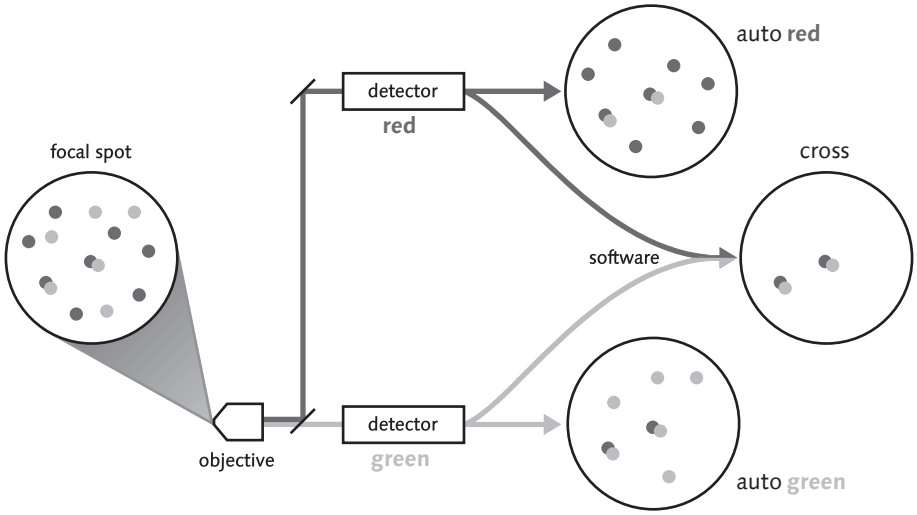

Figure 1. Simplified schematic of the detection principle. Particles with red, green and combined red/green fluorescence are detected in a small confocal volume (illuminated by two lasers). Red and green fluorescence are detected in separate channels. Application of the auto-correlation function produces information on particles exhibiting either red or green fluorescence. Cross-correlation exclusively yields information on particles detected in both channels. Adapted from Schwille *et al.* (1997).

(Schwille *et al.*, 1997) circumvents the need for a mass difference between the binding partners (see figure 1).

In this study, we report the detection of single molecules of DNA performing a computation. Our procedure for experimental implementation relies on the blocking algorithm (Rozenberg & Spaink, 2003), a parallel search methodology which involves direct inactivation of those molecules that are not a solution. Fluorescence cross-correlation spectroscopy was employed to detect hybridization between single DNA molecules. We have tested this technology on a small instance of the NP-complete Satisfiability problem.

Materials and methods

Sequence design

The library for a four variable SAT problem (2^4 possible solutions) was encoded by 16 different oligomers of 36 nucleotides each. Sequences and considerations in sequence design are identical to those in chapter 2 (CEL I and heteroduplex migration assays). Briefly, library strands have the structure

$$5' \; [start][a][b][c][d][stop],$$

where *start* and *stop* are a leader and end sequence, CTTGCA and TTGCAC, respectively, and *a*, *b*, *c*, and *d* stand for the four different variables of the SAT problem. For each of these variables identical subsequences were used to encode its value, ATCACC for 0 (*false*), and GTCTGA for 1 (*true*). Which variable is encoded by one of these two different bit sequences is determined by its position on the DNA strand. Falsifying molecules (blockers) are designed to be complementary to the library oligonucleotides:

$$3' \; [start][a][b][c][d][stop],$$

using subsequences (3'→5') TAGTGG and CAGACT for values 0 and 1, respectively. Library oligos are named after the binary number they encode, for example the oligo corresponding to *abcd* = {0100} is called 08. Blockers are given letter names for the clause they falsify, with the first clause corresponding to A, and so on. Multiple blocker molecules are possible per clause, so a number (0 or 1) is appended to the letter name.

Oligonucleotide synthesis and fluorescent labelling

Custom synthesized DNA oligonucleotides were purchased from Isogen Bioscience (Maarsen, The Netherlands) and IBA (Göttingen, Germany). Library oligonucleotides were covalently labelled with Cy5 (Amersham Biosciences, Piscataway, NJ) at their 5' end (Isogen Bioscience), blocker oligonucleotides with Rhodamine Green (Molecular Probes, Leiden, The Netherlands) at their 5' end (IBA). Each oligonucleotide was purified by denaturing gel electrophoresis (12% polyacrylamide) to remove unbound dye as well as failure sequences. The purified oligonucleotides were diluted in water and their concentration was estimated from the absorption spectra, calculating the molar absorption coefficients at 260 nm according to Sambrook & Russell (2001). For estimating the dye concentration absorption coefficients of $\varepsilon_{647\,nm}$ = 250,000 $cm^{-1}\,M^{-1}$ (Cy5) and $\varepsilon_{500\,nm}$ = 54,000 $cm^{-1}\,M^{-1}$ (Rhodamine Green) were used.

Hybridization assay

For hybridization experiments equal amounts of oligonucleotides were mixed at a concentration of 10 nM each. The mixture was heated to 90 °C for 2 minutes and cooled to room temperature. For the short oligonucleotides utilized in this study, the rate of cooling was found to be irrelevant for the amount of hybridization. All experiments were performed in 1× SSC buffer (150 mM NaCl, 15 mM sodium citrate, pH 7.0) at room temperature (20 °C). Sodium hydroxide was added as indicated in the text in order to prevent mismatch hybridization by lowering the melting temperature. Since leader and end sequences are identical for all library and blocker oligonucleotides, we assume that any effect of the fluorescent dyes on DNA duplex stability will be the same for all combinations.

Theory of fluorescence correlation spectroscopy

FCS (Madge *et al.*, 1972; Eigen & Rigler, 1994) was used to analyse the fluorescence intensity fluctuations originating from single fluorescent labeled DNA molecules diffusing in a confocal detection volume of <0.5 fl. Correlation of the intensity fluctuations over time yields the so-called autocorrelation function, $G(t)$. Using a one component model the experimental autocorrelation curves were fitted by:

$$G(t) = 1 + \frac{1}{N} \cdot \frac{1 - t + T \cdot e^{(-t/\tau_t)}}{\left(1 + \frac{t}{\tau_{diff}}\right) \cdot \sqrt{1 + \frac{t}{SP^2 \cdot \tau_{diff}}}},$$

where N denotes the number of fluorescent particles in the detection volume, T the fraction of fluorophores in triplet state, τ_t the triplet lifetime, τ_{diff} the diffusion time and SP the structural parameter describing the confocal volume.

In dual-colour cross-correlation spectroscopy (Schwille *et al.*, 1997; figure 1) a sample containing two fluorescent species labelled with two different dyes is excited simultaneously with two laser lines, and the fluorescence signals from the two dyes are detected separately. In addition to the autocorrelation functions, the cross-correlation function, $G_{cc}(t)$, is calculated. The latter was fitted according to:

$$G_{cc}(t) = 1 + \frac{1}{N_x} \cdot \frac{1}{\left(1 + \frac{t}{\tau_{diff,gr}}\right) \cdot \sqrt{1 + \frac{t}{SP^2 \cdot \tau_{diff,gr}}}} \cdot$$

This equation uses an apparent particle number, N_x, which is the inverse amplitude of the cross-correlation function:

$$N_x = \frac{1}{G_{cc}(0) - 1}.$$

The diffusion behaviour of the double labelled molecules is described by $\tau_{diff,gr}$. Assuming that there is no cross-talk between the two detection channels (Schwille *et al.*, 1997), the number of doubly labelled particles, N_{cc}, may be determined from:

$$N_{cc} = \frac{N_{ac,r} \cdot N_{ac,g}}{N_x}.$$

Here, $N_{ac,r}$ and $N_{ac,g}$ are the particle numbers obtained from the autocorrelation functions for the red and the green detection channels, respectively.

Cross-correlation measurements

Dual-colour fluorescence cross-correlation measurements were performed on a ConfoCor2 (Zeiss, Jena, Germany), using the cross-correlation configuration (Schwille *et al.*, 1997; Bacia *et al.*, 2002). Dual-colour excitation was achieved with the 488 nm and 633 nm laser lines. The fluorescence emission was split by a dichroic mirror (secondary beam splitter 635 nm), passed through a 505–550 nm bandpass and a 650 nm longpass filter, respectively, and recorded in two separate channels. Fluorescence fluctuations were detected using two avalanche photodiodes. The signals were software-correlated to obtain the autocorrelation and cross-correlation functions. Calibration measurements with standard dyes (Rhodamine6G and Cy5) were performed to determine the geometry and the size of the detection volumes in both channels. Typically, volumes of 0.12 fl and 0.43 fl were found for the green and red channels, respectively. To maximize the overlap of the two detection volumes, pinhole alignment was performed as described by Bacia *et al.* (2002). All measurements were carried out in 10 µl volumes in eight-well glass bottom chambers (Nunc GmbH, Wiesbaden, Germany). For each cross-correlation curve, five individual measurements (30 s each) were averaged. The ConfoCor2 software was used for fitting autocorrelation and cross-correlation curves and to calculate the average number of doubly labelled molecules in the detection volume.

Results

DNA computation

SAT problems have frequently been tackled by DNA-based computations and may be considered a benchmark for new algorithms (see chapter 1). The specific SAT instance solved here is a four variable 3SAT problem consisting of four clauses:

$$F = (\neg a \vee b \vee \neg c) \,\&\, (a \vee \neg b \vee d) \,\&\, (\neg a \vee c \vee \neg d) \,\&\, (b \vee c \vee \neg d),$$

where *a*, *b*, *c* and *d* are the four variables with values of *true* (1) or *false* (0). *or* operations are denoted by '∨', *and* operations by '&', while '¬' symbolizes the negation of a variable. Since the clauses are connected by *and*, falsifying one clause is sufficient for falsification of the complete formula.

Our experimental algorithm (Rozenberg & Spaink, 2003) works as follows: first, all library molecules are synthesized, i.e. a mixture of single-stranded (ss) DNA oligonucleotides encoding all candidate solutions for a given problem. Then, a set of so-called blocker oligonucleotides is created which encode the falsifying assignments for each of the clauses. These falsifiers or blockers are used to block those library molecules that are not a solution. Their sequence is chosen to be complementary to the corresponding library molecules. Addition of the blockers to the library molecules results in hybridization of blockers to all 'wrong' assignments. Hence, the remaining ssDNA molecules are those that do represent a solution. Double-stranded (ds) DNA, representing non-satisfying assignments, is recognized using hybridization detection as output. In order to obtain an addressed array for the output, the library molecules (2^n for a problem with n variables) may either be immobilized on a surface (DNA chip) or distributed in 2^n different tubes.

Several techniques were previously tested for the experimental implementation of the blocking algorithm (described in chapter 2). However, these methods (all employing oligonucleotide concentrations in the micromolar range) did not always allow clear discrimination of all satisfying solutions from all non-satisfying ones.

This study reports the use of single-molecule fluorescence spectroscopy for hybridization detection. First, the development of an assay for hybridization detection at the level of single molecules will be described. Optimization of the experimental conditions allowed hybridization detection with addition of multiple blocker oligonucleotides simultaneously (in parallel). This approach was used to solve the four variable, four clause 3SAT problem described above.

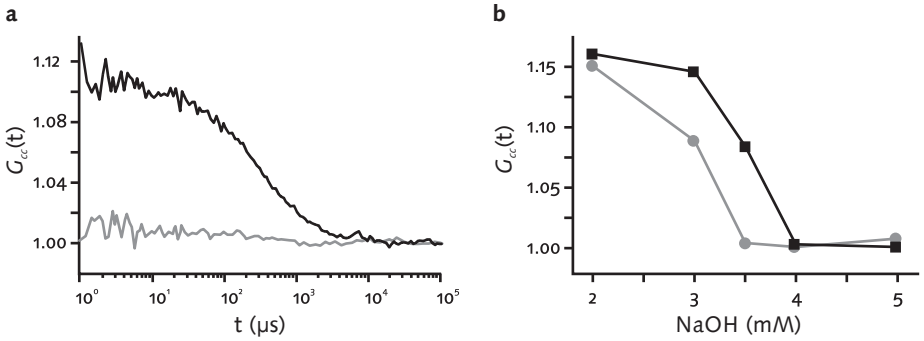

Figure 2. Hybridization detection with single molecules. **a** Cross-correlation curves of two different combinations of library and blocker oligonucleotides. Black line: library 04 and blocker B0, perfect match. Grey line: library 05 and blocker B0, mismatch of four basepairs. All oligonucleotide concentrations were 10 nM (in 1× SSC buffer). 3.5 mM NaOH was added for stringent conditions. Both traces are the average of 5 individual measurements. **b** Effect of NaOH concentration on the initial amplitude of the cross-correlation curve, $G_{cc}(0)$. Black squares: library 04 and blocker B0, perfect match. Grey circles: library 05 and blocker B0, mismatch of four basepairs.

Hybridization detection at the level of single DNA molecules

In order to detect hybridization of single DNA molecules we applied dual-colour fluorescence cross-correlation spectroscopy. For this purpose, the library molecules were covalently 5' labelled with a red fluorescent dye (Cy5) and blockers with a green fluorescent dye (Rhodamine Green). Hybridization of a blocker molecule to a library molecule results in the formation of a dsDNA molecule which is labelled with both the red and the green dye. Since the cross-correlation function contains only dynamic information about doubly labelled molecules (Schwille *et al.*, 1997), a cross-correlation signal is not detected unless formation of dsDNA (hybridization) occurred. Figure 2a depicts two different cross-correlation experiments using two different library molecules (04 and 05) to which the same blocker molecule (B0) was added. Blocker B0 and library 04 form a perfect duplex, whereas hybridization with library 05 results in four mismatched basepairs.

As expected, a cross-correlation signal was only observed for the perfect duplex (blocking combination); the amplitude detected for the four basepair mismatch combination is nearly 0, $G_{cc}(t) = 1$. However, this clear difference is only observed under stringent conditions. At room temperature, without the addition of denaturing chemicals (sodium hydroxide, urea or formamide), the amplitude of the signal for the mismatch combination is considerably higher, indicating the occurrence of mismatch hybridization. Sodium hydroxide (or more general,

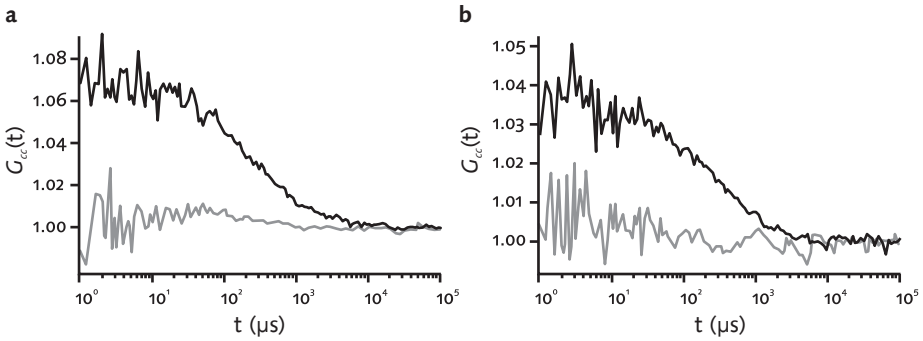

Figure 3. Addition of multiple blocker oligonucleotides in parallel. **a** Cross-correlation curves for hybridization with two blockers. Black line: library 11 plus blockers A0 and A1 (perfect match). Grey line: library 03 plus blockers A0 and A1. **b** Cross-correlation curves for hybridization with three blockers. Black line: library 04 plus blockers B0 (perfect match), B1 and C1. Grey line: library 05 plus blockers B0, B1 and C1. All oligonucleotide concentrations were 10 nM (in 1× SSC buffer with 3.5 mM NaOH).

alkaline conditions) lowers the thermal melting point of DNA (Williams *et al.*, 2001). The effect of NaOH on hybridization is shown for both combinations in figure 2b. The largest difference between the amplitudes for the perfect duplex and the four bp mismatch was observed after addition of 3.5 mM NaOH, indicating that these conditions lower the melting temperature of mismatch combinations to <20 °C. Therefore, in all subsequent experiments 3.5 mM NaOH was added. It is worth mentioning that the optimal amount of NaOH needed for discrimination between perfect duplex and mismatch hybridization depends on the buffer composition, the temperature as well as the length and sequence of the oligonucleotides (data not shown). The latter two parameters can be neglected for our experiments because we designed our sequences to have isothermal melting characteristics.

Addition of multiple blockers in parallel

An important goal for the experimental implementation of the blocking algorithm is the addition of multiple blockers in parallel. Figure 3a illustrates hybridization experiments with addition of two different blocker oligonucleotides at the same time. Again, two different experiments using two different library molecules are compared. Library 11 forms a perfect duplex with blocker A1 and a four bp mismatch with blocker A0, whereas library 03 has a mismatch of four bp and eight bp with blocker A0 and A1, respectively. Like in the experiments with just one blocker added, a cross-correlation signal was only observed for the perfect duplex but not for the mismatch combination. Similar experiments using three

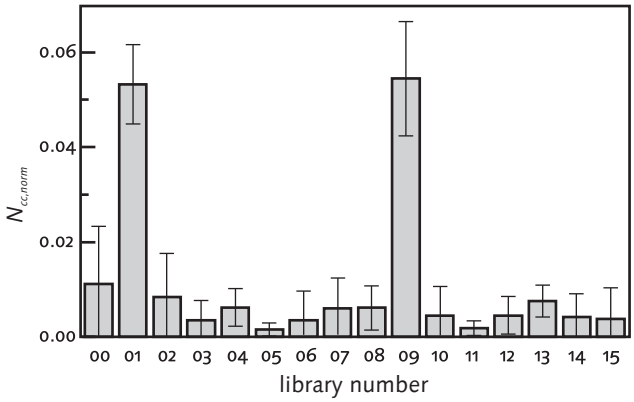

Figure 4. Solution of clause ($b \vee c \vee \neg d$), falsified by *abcd* = {0001} ∨ {1001}. Cross-correlation curves were measured for the 16 different library oligonucleotides after hybridization with blocker oligonucleotides C0 and D0. The average number of doubly labelled molecules, N_{cc}, was determined by fitting the average of five cross-correlation curves for each library/blocker combination. In order to compare experiments from different measurement series, all values were normalized for the same number of blocker molecules, yielding $N_{cc,norm}$. To test the reproducibility of the approach, the results from three to four individual experiments were averaged; the error bars give the standard deviations.

blockers in parallel are shown in figure 3b. Library 04 perfectly matches blocker B0 and has mismatches of four bp and eight bp with blocker B1 and C1, respectively. Library 05 has a mismatch of four bp with blockers B0 and C1, and of eight bp with blocker B1. Again, a cross-correlation signal is only detected for the blocking combination (perfect duplex) and the amplitude observed for the mismatch combination is close to 0.

These experiments demonstrate that the cross-correlation technique is suited for specific hybridization detection in a high background of competing oligonucleotides. However, our data indicate that the amplitude of the signal for the perfect match decreases with an increasing number of blocker oligonucleotides added. In agreement with this observation, the amplitudes observed in experiments using four different blocker molecules in parallel were too low for reliable hybridization detection (data not shown).

Complete computation

To test the applicability of the single-molecule approach for a complete computation, the last clause of the four clause SAT problem described above was used, ($b \vee c \vee \neg d$), which is falsified by *abcd* = {0001} ∨ {1001} (encoded by blockers C0 and D0). In order to compare the amount of hybridization, the average number

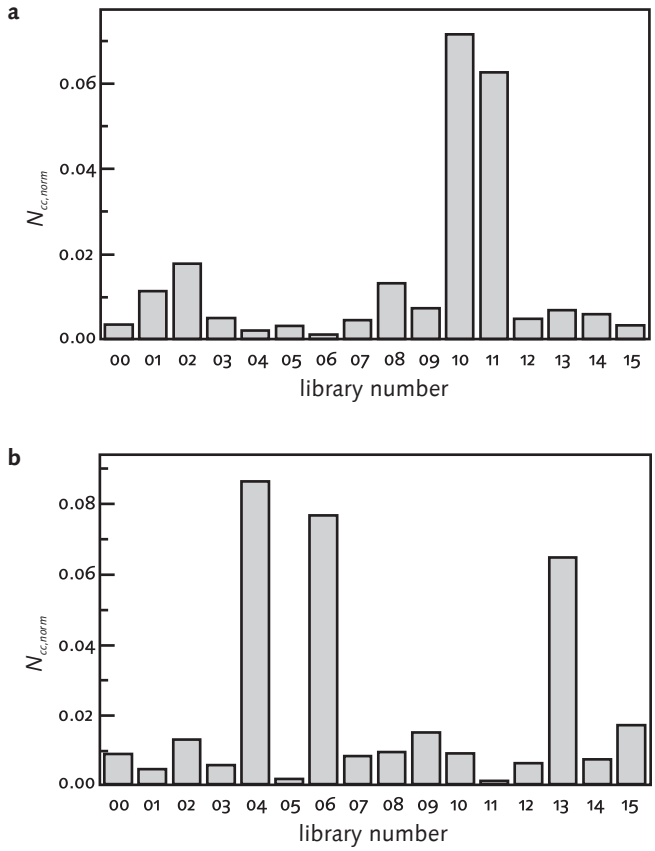

Figure 5. Solution of the remaining clauses of problem *F*. **a** ($\neg a \vee b \vee \neg c$), falsified by {1010} $\vee$ {1011}. Cross-correlation curves were measured after hybridization with blockers A0 and A1. **b** ($a \vee \neg b \vee d$) & ($\neg a \vee c \vee \neg d$), falsified by {0100} $\vee$ {0110} $\vee$ {1001} $\vee$ {1101}. Cross-correlation curves were measured after hybridization with blockers B0, B1, and C1. Blocker C0 was omitted from the experiment since it was already used for falsifying clause ($b \vee c \vee \neg d$), see figure 4. Again, the results of three to four individual measurements were averaged (error bars not shown). The average number of doubly labelled molecules was determined as described for figure 4.

of doubly labelled molecules in the detection volume, N_{cc}, was determined from the experimental cross-correlation curves. Figure 4 shows the results for the 16 different library molecules. Significant numbers of doubly labelled molecules, implying hybridization with blockers, were only detected for library molecules 01 and 09. Library 01 forms a perfect duplex with blocker D0, while library 09 perfectly matches to blocker C0. Independent experiments with the same blocker and library molecules indicate a rather good reproducibility (see error bars in figure 4) and reliable distinction of blocking versus non-blocking combinations.

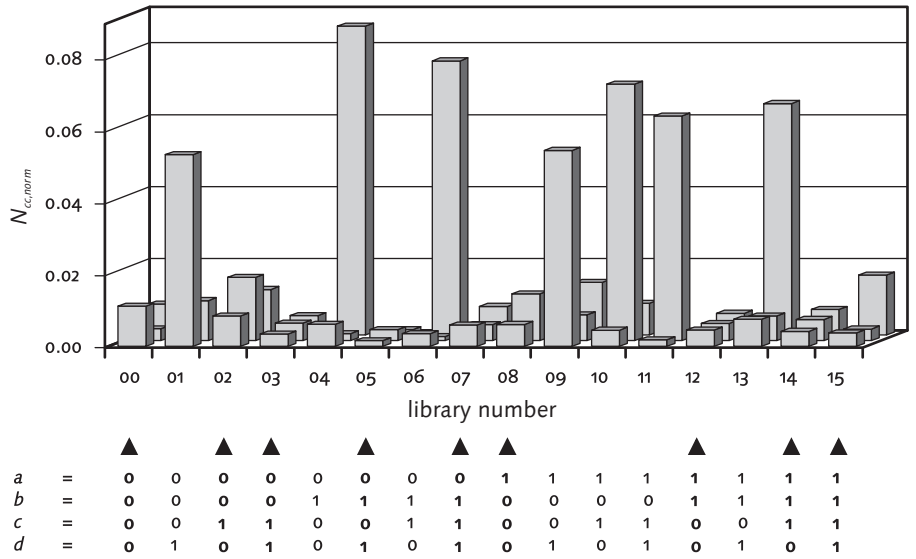

Figure 6. Solution of a four clause, four variable SAT problem, $F = (\neg a \vee b \vee \neg c)$ & $(a \vee \neg b \vee d)$ & $(\neg a \vee c \vee \neg d)$ & $(b \vee c \vee \neg d)$. Summary of the three experiments depicted in figures 4 and 5. The arrows indicate assignments that satisfy the problem instance.

The first clause of formula, $(\neg a \vee b \vee \neg c)$, is falsified by blockers A0 and A1. The corresponding experiment is depicted in figure 5a. Again, a significant amount of doubly labelled molecules was only observed for two combinations. Library 10 and 11 form perfect matches with blockers A0 and A1, respectively. Figure 5b illustrates the results for solving the third and fourth clause of the problem, $(a \vee \neg b \vee d)$ & $(\neg a \vee c \vee \neg d)$. This conjunction is falsified by blockers B0, B1, C0 and C1. Since blocker C0 was already used in the first experiment, and the corresponding library molecule was already disqualified as a solution to F, it may be omitted from the experiment. Experiments with the three other blockers resulted in high cross-correlation signals, i.e. considerable numbers of doubly labelled molecules, for library molecules 04, 06 and 13.

The complete four clause SAT instance is solved by combination of the results of the three previous experiments. Every library molecule that shows a cross-correlation signal in one of the experiments hybridizes to one of the blockers, and is therefore identified as an incorrect assignment. Figure 6 shows that these are library molecules 01, 04, 06, 09, 10, 11 and 13. Formula F is therefore satisfied by the remaining library molecules 00, 02, 03, 05, 07, 08, 12, 14 and 15.

Discussion

The potential of single molecule techniques was already recognized immediately following the first experiments in DNA computing (Adleman, 1996; Reif, 1998). Adleman (1996) proposed that the use of single-molecule fluorescence spectroscopy could enable DNA detection without previous amplification. Our study presents the first example for the utilization of single-molecule techniques for DNA computing. Fluorescence cross-correlation spectroscopy was employed to monitor a DNA computation at the level of single molecules. Given that even single basepair mismatch discrimination can be achieved (data not shown), the technique may also prove to be useful for biomedical applications, e.g. mismatch detection assays or detection of gene expression (Korn *et al.*, 2003) in living cells (Tsuji *et al.*, 2000).

The single-molecule hybridization assay is a considerable improvement compared to the ensemble FRET measurements (chapter 2). Apparently, fluorescence cross-correlation is less prone to experimental errors, most probably because the signal amplitude does not depend on the distance between the two dyes. In this view, hybridization detection would be less susceptible to small differences in DNA structure (which may be caused for example by interaction between nucleobases and the dye molecules; Marras *et al.*, 2002). In addition, the method can also be applied for oligonucleotides longer than the Förster distance of the two dyes.

The DNA computation described here compares the hybridization behaviour of 112 different oligonucleotide combinations. For oligonucleotide sequence design nearest-neighbour thermodynamic parameters were employed (SantaLucia, 1998), which were derived from ultraviolet absorption measurements using micromole solutions of oligonucleotides (SantaLucia *et al.*, 1996). Nonetheless, we could remarkably well predict the hybridization behaviour of all 112 oligonucleotide combinations in the single-molecule hybridization assay. Moreover, our experiments demonstrate that chemical melting with alkaline conditions corresponds very well to thermal melting.

The largest DNA computation reported up to now is the solution of a 20 variable SAT problem (Braich *et al.*, 2002) using gel electrophoresis techniques. The four variable test problem used here is rather small, but in contrast to all previous approaches, DNA computing using single-molecule detection requires neither large quantities of DNA for detection, nor does it destroy the output molecules (Faulhammer *et al.*, 2000; Liu *et al.*, 2000; Sakamoto *et al.*, 2000; Braich *et al.*, 2002). Therefore, the method holds considerable promise for extension of the size of the libraries to be searched, thereby enhancing the performance of parallel search algorithms.

The approach for solving SAT described here and in the previous chapter is

the first experimental implementation of the blocking algorithm described by Rozenberg and Spaink (2003). Opposed to other parallel search algorithms, this methodology sorts out those molecules that encode wrong assignments. In theory only three experimental steps are required: DNA synthesis, hybridization and detection of ssDNA. Our actual computation involved seven steps, synthesis and three iterations of hybridization and detection.

For practical reasons, the experiments described in this study were performed in a volume of 10 µl. Since the detection volume of our setup is <0.5 fl, the amount of DNA may still be significantly reduced. State-of-the-art spotting techniques used for lab-on-a-chip technologies enable dispensing of 60 pl in nanolitre wells (Young *et al.*, 2003), which would decrease the amount of DNA needed for computation to as little as 0.6 attomole of each species. Compared to gel electrophoresis with fluorescent staining (Tuma *et al.*, 1999), which is the most commonly applied detection method for DNA computing, a reduction by four orders of magnitude can be achieved. Thus the search space may be increased 10,000 times.

Hybridization detection by single molecule fluorescence spectroscopy is also much faster than gel electrophoresis. All experiments for solving a four clause SAT instance can be performed in the laboratory in just a single day. FCS can easily be combined with high-throughput screening, meaning that the speed of the computation can be further increased by automation of the procedure and even larger problems may be tackled with affordable computing time. Sophisticated temperature controlled capillary systems may be used to implement high throughput hybridization detection and possibly will allow the addition of more than three blockers in parallel. In addition, the utilization of universal nucleotides may decrease the number of blocker molecules needed for the computation (Loakes, 2001).

The combination of single-molecule fluorescence spectroscopy with lab-on-a-chip technology appears to be especially attractive for evolutionary DNA computing, that is the implementation of evolutionary algorithms with DNA (Chen & Wood, 2000; Bäck *et al.*, 2003). So far, no successful experimental realization of this approached has been reported. In particular the selection step is difficult to implement: selection of the fittest molecules requires very accurate detection (and even sorting) of low quantities of molecules with desirable properties, while a very high background of 'wrong' molecules is present at the same time. One can envisage that single molecule technology will enable selection and manipulation of single desirable molecules for this purpose. Furthermore, the non-destructive character of single-molecule detection allows for iterated selection cycles.

In summary, the application of single-molecule techniques has the potential to overcome some of the current limitations of DNA computing and to extend the scale of computations from proof-of-principle towards real applications.

4

Protein output for DNA computing

Based on:

C.V. Henkel, R.S. Bladergroen, C.I.A. Balog, A.M. Deelder, T. Head, G. Rozenberg & H.P. Spaink (2005) Protein output for DNA computing. *Natural computing, in press*

Abstract

An important area of research in DNA computing is the detection and analysis of output molecules. We demonstrate how DNA computing can be extended with *in vivo* translation of the output. The information per mass unit is about 15-fold higher in the resultant proteins than in the original DNA output. The proteins are therefore of correspondingly lesser mass, which facilitates their subsequent detection using highly sensitive mass spectrometry methods.

We have tested this approach on an instance of the Minimal Dominating Set problem. The DNA used in the computation was constructed as an open reading frame in a plasmid, under the control of a strong inducible promoter. Sequential application of restriction endonucleases yielded a library of potential solutions to the problem instance. The mixture of plasmids was then used for expression of a protein representation. Using MALDI-TOF mass spectrometry, a protein corresponding to the correct solution could be detected. The results indicate the feasibility of the extension of DNA computing to include protein technology. Our strategy opens up new possibilities for both scaling of DNA computations and implementations which employ output of functional molecules or phenotypes.

Introduction

An important problem in biomolecular computing is the generation and analysis of output molecules. Here, we have exploited the natural capacity of DNA to direct the synthesis of proteins, which can be used as output molecules. An advantage of the use of proteins for molecular computations is that much higher information densities are possible than using nucleic acids: using translation, the information content of a DNA triplet (approximately 2000 Da) is expressed in one 57–186 Da amino acid. The small proteins can then be accurately analysed using modern proteomics technology, where the original DNA molecules would be too bulky to examine by mass spectrometry.

A computation was conducted on a DNA sequence constituting an open reading frame (ORF), which was placed under the control of a strong promoter (figure 1). This enables the in vivo transcription and translation of the computational construct into a protein. Mass spectrometry then allows sensitive determination of both size and composition of the expressed library in parallel.

We have tested the principle of protein output on an instance of the Minimum Dominating Set (MDS) problem, using plasmid DNA as computing hardware (Head, 2000; Head *et al.*, 2000). In this approach, restriction endonucleases are used to specifically remove 'stations' from a computing plasmid (figure 1). The absence or presence of these stations in the plasmid, which is basically a

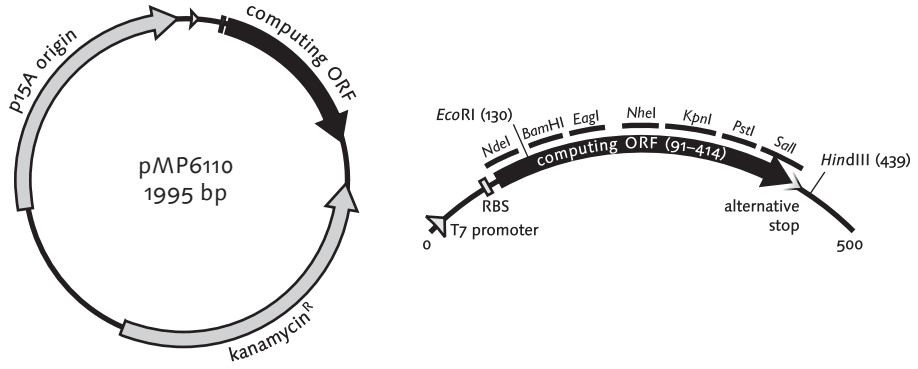

Figure 1. The computing region of computational plasmid pMP6110. Information is present at different levels: a 'station' in the plasmid has a certain length on DNA level, and the associated peptide has a certain weight. Sequences have been chosen such that DNA length corresponds to protein weight. Additionally, the DNA sequences encode known protein epitopes. For these, antibodies are commercially available, enabling an alternative detection system. One of these peptide tags (HA) is used for purification of the expressed protein library and is not used in the computation. Stations are named after the enzymes that can excise them.

computer memory, corresponds to a bit set to either 0 or 1.

The computation starts with an aqueous solution of a single species of plasmid. Because of the fluid medium and the high number of plasmids, it can be assumed that splitting the mixture in several distinct volumes yields as many identical copies of memory. These memories are then modified by application of certain restriction endonucleases and religation (removal of stations). This enzymatic 'software' acts on millions of plasmids in parallel. After memory writing, the subsets are mixed again. Iteration of this procedure results in a library containing an exponential number of plasmids. DNA representing a solution to a given computational problem can be identified by selection on certain characteristics, for example length. Plasmid computing was successfully used to solve a six variable instance of the NP-complete Maximal Clique problem (Head *et al.*, 2000) and can be adapted to a broad range of algorithmic problems (Head *et al.*, 2002b).

Materials and methods

Computational plasmid

Plasmid pMP6110 is a length-minimized derivative of pOK12 (Vieira & Messing, 1991; Head *et al.*, 2000), containing an *E. coli* origin of replication and a kanamycin resistance marker. The computing open reading frame (ORF) is under the control of the strong inducible T7 promoter and a consensus ribosome binding sequence (RBS). Stations in the ORF (see table 1) encode the following protein epitopes: HA, 8× His-tag and c-myc (Roche Diagnostics); FLAG (Sigma-Aldrich); S-tag (Novagen); TE (thrombin and enterokinase cleavage sites); *enod40* (Staehelin *et al.*, 2001). The ORF translation stop is located inside the last station (position 415). If this station is removed, an alternative stop codon is located 15 basepairs downstream (430). The net effect is a relatively low peptide mass associated with the *Sal*I station (see table 1). Synthetic oligonucleotides used in plasmid construction were purchased from Isogen Bioscience (Maarssen, NL).

Table 1. Information levels in plasmid pMP6110

Station	Excision enzyme	DNA length	Peptide mass	Epitope
HA	*Nde*I	36 bp	1.41 kDa	HA
b	*Bam*HI	36 bp	1.36 kDa	8x His
e	*Eag*I	36 bp	1.44 kDa	c-myc
n	*Nhe*I	36 bp	1.27 kDa	FLAG
k	*Kpn*I	51 bp	1.89 kDa	S-tag
p	*Pst*I	36 bp	1.32 kDa	thrombin/enterokinase
s	*Sal*I	45 bp	1.13 kDa	*enod40*

Library generation

An initial plasmid quantity of 40 ng was sequentially digested and religated as shown in figure 2b. Enzymes were purchased from New England Biolabs and handled according to the manufacturers recommendations. After all enzymatic reactions, the reaction mixture was purified using QIAquick PCR cleanup kits (Qiagen). Ligations were carried out overnight at 16 °C in a 400 µl reaction volume. This combination of fragment removal and large volume minimizes the likelihood of religation of the excised station. After ligation, the plasmid mixture was transformed into *E. coli* XL1-blue cells for amplification and isolated again using QIAprep plasmid miniprep kits (Qiagen). The resulting library was analysed using polyacrylamide gel electrophoresis and bands were visualized using SYBR Green (Molecular Probes, Leiden, NL).

Protein purification

Protein was purified from *E. coli* BL21(DE3) (Invitrogen) induced with isopropyl-β-D-thiogalactopyranoside (IPTG). Cells were lysed in 8 M urea, extract was dialysed against 10 mM Tris pH 8, 1 mM EDTA and tagged proteins were purified using an anti-HA affinity column (Roche Diagnostics). Purified protein was concentrated using Microcon YM-3 concentrators (Millipore). Gel electrophoresis and staining were performed as described (Schagger & von Jagow, 1987; Sambrook & Russell, 2001).

Mass spectrometry

Protein samples were desalted using Centri Spin 10 gel filtration columns (emp Biotech, Berlin). Spectra were recorded on a Bruker Reflex III mass spectrometer in linear mode, using 2,5-dihydroxybenzoic acid supplemented with fucose as a matrix. The identity of the original pMP6110 protein product was confirmed by analysis of trypsin and chymotrypsin digests on a Bruker Ultraflex (data not shown).

Results

Minimal dominating set

Given a graph with vertices (nodes) and edges (connections), the MDS problem asks for the smallest possible vertex set from which all other vertices can be reached by an edge. The graph used is shown in figure 2a.

The MDS problem is a representative of the large and important class of NP-complete problems and is as such equivalent to all other problems in this class (Garey & Johnson, 1979). Other instances of NP-complete problems that have been used to test DNA computing approaches include Directed Hamiltonian Path, Maximal Clique and Satisfiability problems (chapter 1). In particular, the six node MDS problem considered here is related to a six variable, six clause Satisfiability problem.

The algorithm used to arrive at a dominating set exploits the fact that any vertex must be either in the set or in the immediate (one edge) vicinity of the set. The problem can then be restated in terms of the neighbourhoods, $N(v)$, for the vertices v of the graph. A dominating set must contain at least one vertex from each $N(v)$. For example, in figure 2a, neighbourhood $N(p)$ contains b, n, e and p. A dominating set meets the requirements imposed by all six neighbourhoods.

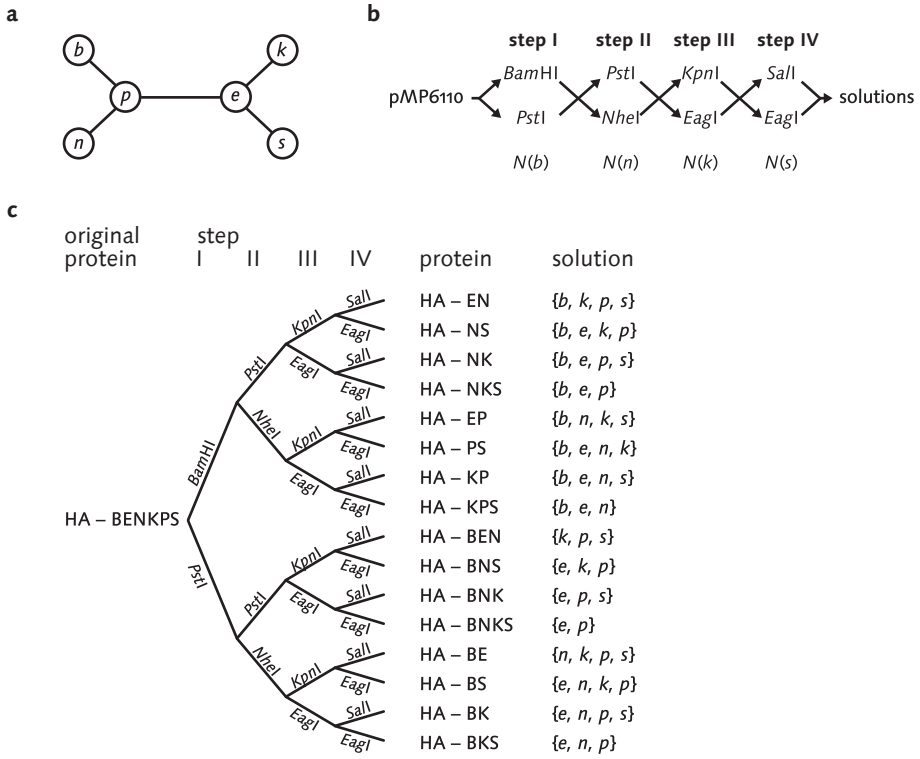

Figure 2. Problem instance and solution strategy. **a** Schematic representation of the graph used. Vertices are named after the available 'stations' on plasmid pMP6110. The graph can be defined as an undirected graph $G = (V, E)$, with V the set of vertices and E the set of edges. Alternatively, the graph can be defined by a set of neighbourhoods: a neighbourhood $N(v)$ for vertex v is defined as the set $N(v) = \{u \text{ in } V\text{: either } u{=}v \text{ or } [u, v] \text{ is in } E\}$. The graph shown here is then given by $N(b) = \{b, p\}$, $N(e) = \{e, k, p, s\}$, $N(n) = \{n, p\}$, $N(k) = \{e, k\}$, $N(p) = \{b, e, n, p\}$, $N(s) = \{e, s\}$. A subset of V is a dominating set precisely in the case that it contains at least one vertex from every neighbourhood. **b** Generation of all possible solutions by digestion and ligation. Only four steps are necessary, since neighbourhoods $N(e)$ and $N(p)$ are redundant. For example, $N(e)$ contains $N(k)$ and $N(s)$, and is therefore already accommodated in step III or step IV. **c** The complete library generated by the procedure shown in figure 2b.

Finding just any dominating set is easy, as the original set of all six vertices already contains at least one member of every neighbourhood. Finding the minimal dominating set, however, requires an exhaustive search of all possible dominating sets.

Experimental algorithm

The MDS instance described above can be solved experimentally in two stages: first, generate candidate dominating sets; and second, select the minimal dominating set. All potential solutions were generated from plasmid pMP6110 using a mix and split methodology (Head *et al.*, 2000), as illustrated in figure 2b, c. The initial, complete plasmid represents an empty subset. The absence of a station from the plasmid is interpreted as the presence of the corresponding vertex in a subset. All required neighbourhoods are accommodated sequentially. For any neighbourhood, the mixture containing all plasmids is divided in as many test tubes as there are vertices in the neighbourhood. In each of those test tubes, the removal of one specific station is assured by restriction digestion and religation.

After four steps, a library of 16 different plasmids was obtained (figure 3a). This mixture was transformed to a suitable *E. coli* host strain for protein overexpression (figure 3b). Protein gel electrophoresis provides neither the resolution nor the sensitivity needed to positively identify proteins. Therefore, the purified protein was analysed using matrix-assisted laser desorption ionisation time-of-flight (MALDI-TOF) mass spectrometry. The resulting spectrum is a representation of all potential minimal dominating sets, with the heaviest protein corresponding to the minimal dominating set (figure 3c). The original protein (product of plasmid pMP6110) has an average mass of 12054.14 Da. The heaviest protein in the mixture is detected at a mass to charge ratio of 9292. This matches the product of the computational ORF missing stations *e* and *p*, which has a predicted average molecular weight of 9291.08 Da. The ORF without these stations in turn corresponds to the minimal dominating set {*e*, *p*}.

Discussion

The successful detection of the protein representation of the minimal dominating set shows that mass spectrometry is an attractive read-out strategy for DNA computing. The problem instance solved here is of roughly the same size as molecular computations reported previously (see chapter 1). The method is potentially up scalable: MALDI-TOF mass spectrometry is capable of accurately detecting protein mass ranges exceeding 50 kDa (Blank *et al.*, 2002), which would correspond to about 40 plasmid stations and a protein library of 10^{12} species (2^{40}). In large-scale approaches, further information on protein identity can be obtained by application of tandem mass spectrometry (Chalmers & Gaskell, 2000) or proteolytic cleavage of the solutions. This approach is not limited to the plasmid computing method. If some encoding constraints are taken into account, answer molecules from any nucleic acid based computation method can be translated.

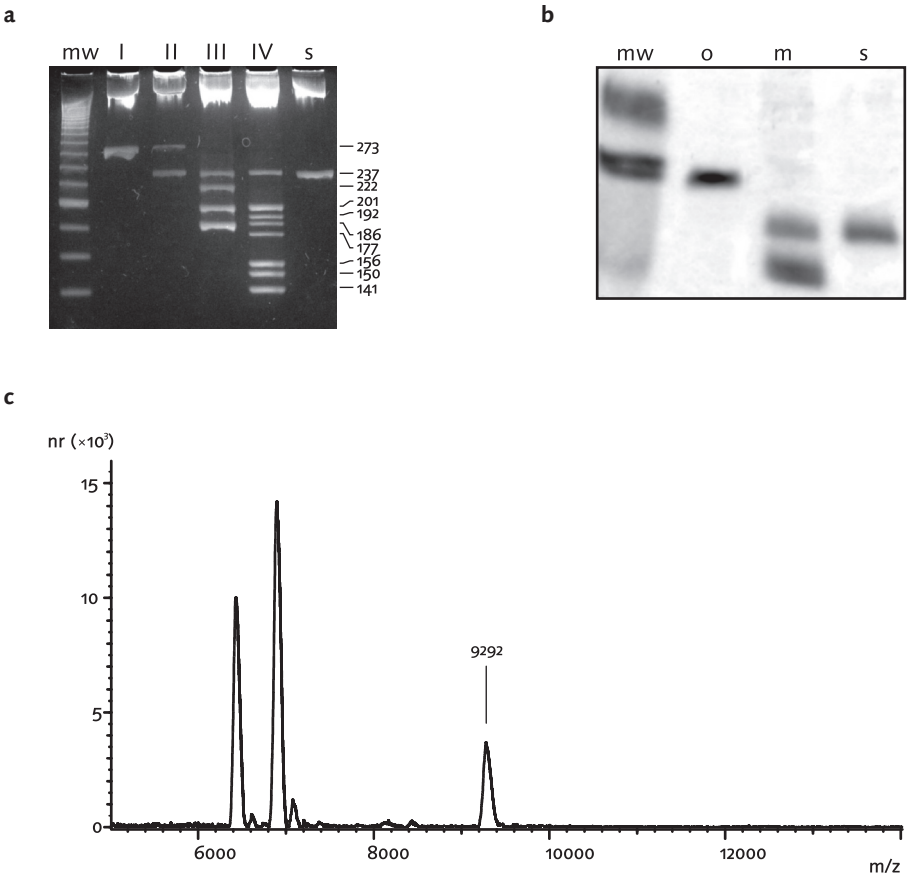

Figure 3. Analysis of potential solutions. **a** 8% polyacrylamide gel with *Eco*RI/*Hin*dIII fragments of religated plasmid after every step (see figure 2b). Lane mw: DNA size marker; lane s: the isolated solution representing set {*e*, *p*}. **b** Silver stained 12% SDS-tricine-polyacrylamide gel with purified protein. Lane mw: molecular weight marker; lane o: original protein (from plasmid pMP6110); lane m: total protein representation (after step IV); lane s: isolated solution {*e*, *p*}. **c** MALDI-TOF mass spectrum of the total protein representation. The y axis shows the number of detection events, the x axis the mass to charge ratio of the detected proteins. Since the charge is predominantly 1, this ratio corresponds to the molecular mass in Daltons. A single protein species contains many different isotopes, and is therefore detected as a mass distribution. The average molecular weight of a molecule is determined by allocating the peak of such a spread. Here, the largest protein detected has a molecular weight of 9292 Da.

In conclusion, the novel output approach for DNA based computing presented here introduces the use of translation. The ribosome is one of the major information processing components of the cell, and is therefore an interesting candidate as a component of artificial biomolecular computers. So far, only one other design for a hybrid DNA/protein computer has been presented (Sakakibara & Hohsaka, 2003). The generation of protein phenotypes offers possibilities for the implementation of evolutionary algorithms in DNA (Chen & Wood, 2000; Bäck *et al.*, 2003). Protein-based computing methods can also employ the potential of the computational output to function as biologically active molecules. For instance, the plasmid used (figure 1) encodes the plant hormonal peptide *enod40* (Staehelin *et al.*, 2001). In this way, the outcome of a computation could act as a biologically active protein, which in turn switches on downstream computational or biological processes.

Acknowledgements

We thank Pascal van der Wegen, Kees Breek, Ron Hokke and Marco Bladergroen for technical assistance and advice.

5

DNA computing of solutions to knapsack problems

Abstract

One line of DNA computing research focuses on parallel search algorithms, which can be used to solve many optimization problems. DNA in solution can provide an enormous molecular library, which can be searched by molecular biological techniques. We have implemented such a parallel search for solutions to knapsack problems, which ask for the best way to pack a knapsack of limited volume. Several instances of knapsack problems were solved using DNA. We demonstrate how the computations can be extended by *in vivo* translation of the DNA library into protein. This combination of DNA and protein allows for multicriterion optimization. The knapsack computations performed can then be seen as protein optimizations, one of the most complex computations performed by natural systems.

Introduction

Plasmid DNA can serve to perform computations at the molecular level. Specially designed plasmids contain a dedicated computing region, which is basically a computer memory with bits set to either 1 or 0. The plasmid memory can be operated on by restriction endonucleases. Removal of a region from the plasmid (a 'station') is identified with the flipping of a bit (Head, 2000). In theory, all plasmids in a solution can represent different memory configurations and therefore provide a huge parallel search space for optimization problems.

Typically, a computation starts with a single species of plasmid, from which a library of different plasmids is generated by repeated modifications to subsets of the plasmid mixture. From this library, a memory configuration corresponding to the solution to a certain problem can be selected using molecular biological separation technologies. Plasmid computing has been successfully applied to small test instances of several computationally hard optimization problems, and can be applied to a broad range of algorithmic problems (Head *et al.*, 2000; Head *et al.*, 2002b).

Recently, plasmid computing was extended with protein expression by the construction of the whole computing region of a plasmid as part of an open reading frame (chapter 4). After library generation, the library was expressed into a protein representation, and this was in turn used to select a solution. A potential advantage of this translation of the solution into protein are smaller molecules (which can be analysed using sensitive mass spectrometry technology) and consequently higher information densities.

In this computation both DNA and protein encoded exactly the same information, i.e. the values of bits. However, the translation process can also be taken

advantage of by specifying different information in DNA and protein. In this way, the optimization of the solution may be realized according to multiple distinct criteria. On the nucleic acid level, DNA length, presence and sequence are obvious encoding possibilities. Amino acids allow for linear representation of these after translation, but add more complex physico-chemical characteristics such a molecular weight and isoelectric properties.

Here, we have solved several instances of knapsack problems using plasmids. These problems ask for the best way to pack a knapsack of limited capacity with items of different size and weight. DNA computing seems very well suited for this family of problems, as both the encoding and the algorithm are relatively straightforward: formal size or weight can be linked directly to physical properties of DNA. In addition, because these problems allow for multi-criterion optimization, both DNA length and protein mass can be used to compute.

Materials and methods

Plasmid construction

pKnapsack1 was constructed by ligation of a synthetic oligonucleotide linker (Isogen Bioscience, Maarssen, NL) into the *Sal*I/*Avr*II of plasmid pMP6110 (chapter 4). The linker replaces the pMP6110 *enod40* region with a VSV-G encoding region (Roche Diagnostics) flanked by *Sal*I recognition sites and a fragment flanked by *Bgl*II recognition sites. Thus, the number of computational stations is increased to eight (see table 1) and the stop codon in the *enod40* encoding region is removed. The next stop codon is 12 bp downstream of the last *Bgl*II site. The entire open reading frame containing the computational stations is under the control of a pET9d derived T7 promoter and a consensus ribosome binding site (figure 1). Integrity of ORF and promoter were checked by DNA sequencing (Baseclear, Leiden, NL). Other features of pKnapsack1, a pOK12 derivative (Vieira & Messing, 1991), are a p15A *E. coli* origin of replication and a kanamycin resistance marker.

Plasmid modification

Stations were removed by digestion of 1–10 µg (determined by absorbance at 260 nm) plasmid with the appropriate enzyme (New England Biolabs) under recommended conditions. Linearized vectors were isolated from a methylene blue stained 1% TAE agarose gel and purified using QIAquick gel extraction columns (Qiagen). Ligations were then performed overnight at 16 °C using T4 DNA ligase (Roche), and the ligated plasmid was transformed to chemically competent *E. coli* XL1-blue (Sambrook & Russell, 2001). This procedure ensures the

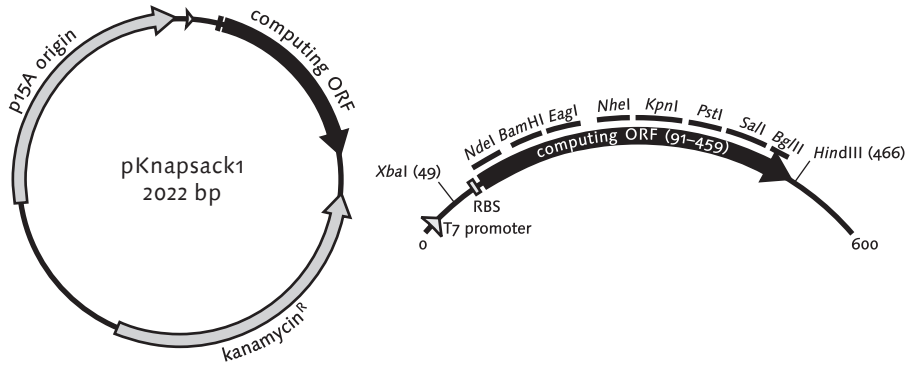

Figure 1. Plasmid pKnapsack1. The computing ORF contains eight stations, each of which can be excised by the indicated enzyme.

Table 1. pKnapsack1 elements

Item	Fragment	Length	Weight	Excision enzyme	Epitope
1	424–445 bp	21 bp	0.55 kDa	*Bgl*II	7× Gly
2	373–418 bp	45 bp	1.74 kDa	*Sal*I	VSV-G
3	331–367 bp	36 bp	1.32 kDa	*Pst*I	thrombin/enterokinase
4	274–325 bp	51 bp	1.89 kDa	*Kpn*I	S-tag
5	232–268 bp	36 bp	1.44 kDa	*Nhe*I	FLAG
6	178–214 bp	36 bp	1.36 kDa	*Eag*I	c-myc
7	88–124 bp	36 bp	1.43 kDa	*Nde*I	HA
	136–172 bp	36 bp	1.41 kDa	*Bam*HI	8× His

removal of the station from all plasmids, and selects for fully functional plasmids. Plasmids were isolated from bacteria using QIAprep plasmid purification kits (Qiagen).

Protein purification

For protein expression, the plasmid mixture was transformed to chemically competent *E. coli* BL21(DE3). This strain carries the T7 RNA polymerase gene under the control of the *lacUV5* promoter. Expressed computational protein was purified by Ni^{2+}/histidine affinity chromatography. Cells were grown to an OD_{600} of 0.5 in LB medium, after which T7 protein expression was induced by addition of isopropyl-β-D-thiogalactopyranoside (IPTG) to a final concentration of 1 mM. After two hours, cells were harvested by centrifugation and lysed in urea buffer (8 M urea, 100 mM NaH_2PO_4, 10 mM TrisCl, pH 8.0). 50 µl Ni-NTA agarose (Qiagen) was added to 500 µl of cleared lysate (corresponding to 5 ml of culture),

and the mixture was incubated with shaking at 4 °C for 30 minutes. The resin was washed twice with 500 µl urea buffer at pH 6.3, and protein was eluted thrice with 50 µl urea buffer at pH 4.5.

Knapsack selection

*Hin*dIII/*Xba*I digested plasmid library was separated on a 25 cm 10% TBE polyacrylamide gel. The minor *Hin*dIII/*Xba*I fragment contains the entire computational region (figure 1). Bands of desired length were cut from the SYBR Gold (Molecular Probes, Leiden, NL) stained gel and isolated by overnight soaking at 4 °C in diffusion buffer (0.5 M NH_4Ac, 0.1% SDS, 2 mM EDTA), centrifugation in 0.22 µm Ultrafree-MC spin filters (Millipore) and ethanol precipitation. Fragments were then religated into the vector (major *Hin*dIII/*Xba*I fragment).

Results

Knapsack problems

The knapsack family of problems asks for ways to pack a volume (knapsack) of limited capacity in the most efficient way. The solution depends on the nature of the items to be packed. In some variants, only size is associated with the items – in others, items have both sizes and values, and the efficiency is evaluated by the total value packed in the fixed size knapsack.

The problem can be defined as follows: Given a set of n items, each with size $s_i \in \mathbb{N}$ (natural numbers including 0) and value $v_i \in \mathbb{N}$, and a knapsack capacity C, maximize

$$\sum_{i=1}^{n} x_i v_i \text{ subject to } \sum_{i=1}^{n} x_i s_i \leq C ,$$

where x_i is the multiplicity of item i in the knapsack. In the binary or 0/1 knapsack problem treated here, the availability of each item is limited to one, so $x_i \in \{0,1\}$. In the unbounded or integer knapsack problem, the supply of each item is unlimited: $x_i \in \mathbb{N}$. If only item size is considered ($s_i=v_i$ for every item i), the binary knapsack problem reduces to the subset sum problem:

$$\sum_{i=1}^{n} x_i s_i \leq C, \;\; x_i \in \{0,1\}.$$

All these problems belong to the class NP-complete, which means that no deterministic algorithm is known to solve them in polynomial time, and it is unlikely such an algorithm exists (Garey & Johnson, 1979). However, algorithms exist that can solve certain knapsack variants in pseudo-polynomial time, i.e. in

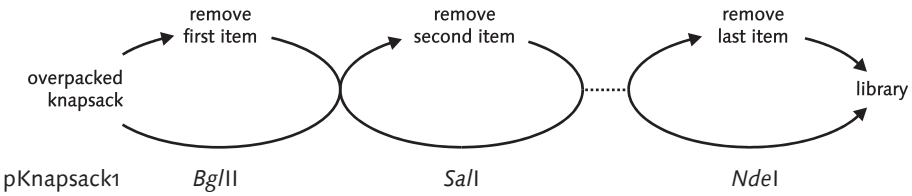

Figure 2. Generation of all possible knapsack fillings by digestion and ligation. For every item, half of the mixture of plasmids is left untreated. In the other half, the station is removed. After this procedure, the two solutions are mixed again and the next item is processed. The separation in two subsets is assumed to yield identical plasmid mixtures.

practice their exponential scaling behaviour can be rather mild. Still, the only guaranteed way to solve instances of the knapsack problem is the exhaustive enumeration of all possible knapsack packings (Garey & Johnson, 1979; Pisinger, 2005).

Molecular algorithm

To implement knapsack problems using plasmids, stations in the computational plasmid (figure 1) are identified with items in the knapsack. Other than in previous computations using a similar plasmid (see for instance chapter 4), presence or absence of the stations is not associated with bit values. Instead, knapsack item sizes are encoded by the length of these stations (the number of basepairs). The items available in plasmid pKnapsack1 are listed in table 1. The experimental algorithm starts with an initial (overstuffed) plasmid knapsack containing all eight items. A library of all possible knapsack packings was generated by the iterated removal of stations from subsets of the plasmid mixture (see figure 2). The station flanked by *Bam*HI sites was not used in the computation. After seven rounds of selective removal, a mixture of 128 different plasmid species was obtained (figure 3a). The plasmid library was then separated according to length by gel electrophoresis, and a knapsack capacity was imposed by excision of a band of the desired size. The most time-consuming step of this algorithm, the library generation, takes $O(n)$ steps for a knapsack with n available items. This is an exponential speedup from the sequential generation of all possibilities, which takes $O(2^n)$ steps.

Subset sum computation

The subset sum problem considers only item size, and items can occur only once. Therefore, a subset can be obtained by excision of a band corresponding to a certain sum from the separated library. For this computation, *Xba*I/*Hin*dIII frag-

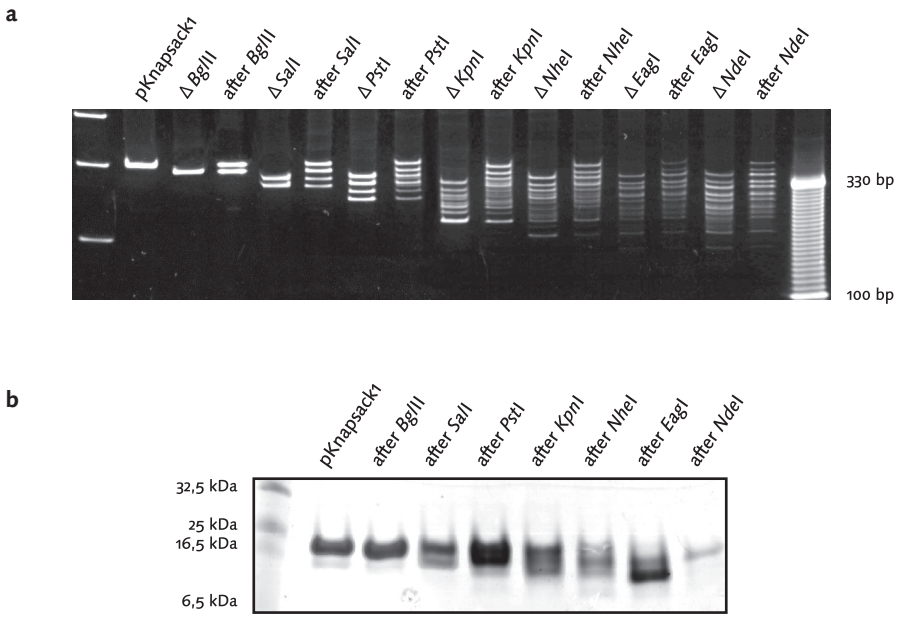

Figure 3. The complete library of potential solutions. **a** Plasmid library. 0.5 µg plasmid was analysed for every station removal and for every subset rejoining. The last lane represents the final library. DNA size markers: Smart Ladder (Eurogentec) and 10 bp ladder (Invitrogen). *Xba*I/*Hin*dIII digests of the plasmids were analysed on 4–12% TBE polyacrylamide gels (Novex precast, Invitrogen) and stained by SYBR Gold (Molecular Probes). Images were captured on a Biorad FluorS Imager using UV excitation and a 530 nm bandpass filter (the figure is a composite of two gel images). **b** Protein library. Protein was purified for every subset joining. 10 µl of the final elution fraction was analysed on a 10% Tris-tricine polyacrylamide gel. The gel was Coomassie stained and scanned on a Biorad GS-800 densitometer. Molecular weight marker: prestained broad range (New England Biolabs).

ments of 260 and 330 basepairs were selected. Because these fragments contain 156 basepairs of additional DNA (see figure 1), this corresponds to knapsacks of C=104 and C=174, respectively. The fragments were isolated from gel, religated into the vector, and introduced into *E. coli*. Individual plasmid species were physically separated by plating of the transformants.

For capacity 104, plasmids from 5 colonies were reisolated (figure 4a) and analysed using restriction enzymes. The first lane contains a plasmid with a 264 bp insert, containing the *Nde*I, *Nhe*I and *Pst*I items. This corresponds to a subset sum of 108. Two plasmids (lanes 2 and 3) were found to contain the *Nde*I, *Sal*I and *Bgl*II stations, summing up to 258 bp (or a subset sum of 102). Two other plasmids (lanes 4 and 5) contain the *Sal*I and *Kpn*I stations, summing up to 252

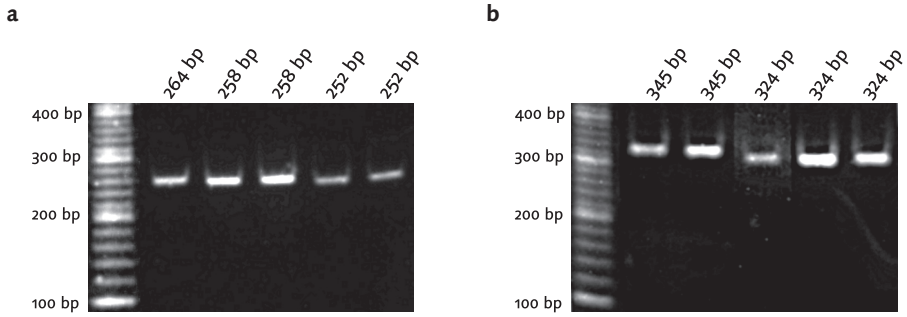

Figure 4. Subset sum computation. *Xba*I/*Hin*dIII digests of five different plasmids with a 260 bp (**a**) or 330 bp (**b**) knapsack capacity imposed, separated on a 4–12% gel and imaged as before. DNA size marker: 20 bp ladder (Sigma).

bp. A fragment of exactly 260 basepairs was not recovered, and exhaustive (non-molecular) enumeration of possibilities indicates that it does not exist.

Capacity 174 was analysed in the same way (figure 4b). Recovered plasmids contained a 345 bp insert (lanes 1 and 2), composed of the *Nde*I, *Kpn*I, *Pst*I, *Sal*I and *Bgl*II stations or the *Nde*I, *Nhe*I, *Kpn*I, *Sal*I and *Bgl*II stations, respectively. Lanes 3, 4 and 5 show 324 bp inserts, composed of either the *Nde*I, *Nhe*I, *Kpn*I and *Sal*I stations (lane 3) or the *Eag*I, *Nhe*I, *Kpn*I and *Sal*I stations. A fragment of exactly 330 bp was not recovered, although it should exist (consisting of *Bgl*II, *Sal*I and three 36 bp stations).

Binary knapsack computation

Because the computational region of the library plasmids is structured as a single ORF under the control of a strong inducible promoter, the computation can be extended to include protein optimization. The molecular binary knapsack problem considered here asks for the maximum protein mass that can be realized for a certain maximum DNA length (analogous to a maximum total value for a certain size capacity). The DNA knapsack capacity was again limited by band excision, however, in this case also knapsack fillings below the maximum capacity were considered. In the presence of heterogeneous items, maximum protein value is not guaranteed by maximum DNA size.

Knapsack capacities were chosen at 144 and 244, corresponding to fragments up to 300 bp and 400 bp, respectively (figure 5a). The mixture of fragments was again religated into the vector and introduced into *E. coli*. The resulting bacterial culture was used for protein expression. Protein was purified from the culture using the affinity of the eight histidine residues present in all computational proteins (encoded by the *Bam*HI station, see table 1) for nickel ions. Figure 3b

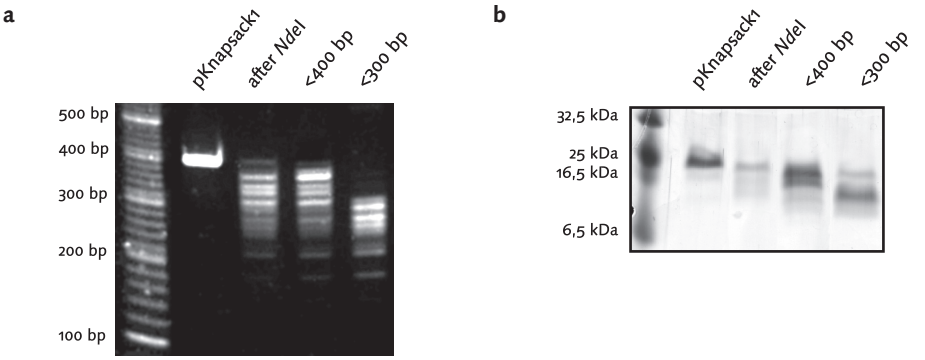

Figure 5. Binary knapsack computation. **a** Size limits: from the original library (after *Nde*I), two pools were generated with maximum sizes of 400 and 300 basepairs, respectively. The largest fragment in the complete library is the original pKnapsack fragment, 417 bp. *Xba*I/*Hin*dIII fragments, imaged as before. **b** Values: translation products from the plasmid pools of figure 5a. Silver stained 10% Tris-tricine polyacrylamide gel, scanned as before.

shows the translated and purified product of the different steps during library construction. Figure 5b shows the translation products of the different knapsacks selected.

The translation product of the pKnapsack1 computing ORF has a predicted mass of 13.5 kDa, a completely empty knapsack yields a protein of less than 4 kDa. Protein gel electrophoresis does not provide the accuracy to characterize such molecules. Figures 3b and 5b can only be used to establish a correlation between knapsack size and value. The identity of the heaviest protein may be determined by mass spectrometry methods, or using antibodies specific for the different epitopes.

Discussion

We have demonstrated how DNA computing can be used to solve small-scale instances of knapsack problems. So far, this family of problems has been largely overlooked by the biomolecular computing field. They have received some theoretical attention, but always in relation to models of computation that have not yet been physically implemented (Chang *et al.*, 2004; Pérez-Jiménez & Sancho-Caparrini, 2002). Two three-item subset sum experiments have been reported, but both approaches appear to have met their practical limits at this size (Aoi *et al.*, 1998; Stoschek *et al.*, 2001). Knapsack problems allow for the most natural encoding in DNA of all molecular computations reported so far. There is no need for elaborate sequence design as seen in models relying on hybridization, and

even the mapping of sequence to formal logic can be dispensed with: the physical dimensions of the DNA used immediately suggest the problem ingredients. Also, the knapsack algorithm makes unprecedented use of DNA's potential parallelism. By separating the plasmid library according to length, many simultaneous knapsack computations take place. In fact, both subset sum capacities enforced in the experiments described here were excised from the same electrophoresis lane.

The use of plasmids is also a crucial factor in the success of this computation. Knapsack items could also be encoded in simple DNA elements, without a vector. However, the plasmid provides better control of the library generation and allows for reliable amplification and storage. The plasmid method itself can also be used in scaling to larger instances, as inserts over 10^4 bp are quite common. If elements on the same scale as used here are inserted, a single plasmid could easily accommodate hundreds of knapsack elements.

A more important factor in scaling is probably the separation technology used. Polyacrylamide gel electrophoresis is, in theory, capable of separating DNA fragments with a resolution of 0.1%, i.e. discrimination between 999 and 1000 basepair fragments is feasible (Sambrook & Russell, 2001). However, these figures are for dedicated full length sequencing gels and capillary sequencing systems. Here, 25 cm gels were used in the selection phase, with the entire fragment library separated over approximately 10 cm. The smallest bands that can be excised are of the order of 1 mm long. Consequently, exact numerical solutions are not to be expected here, and indeed fragments differing up to 21 basepairs were recovered from the same slice. Still, as an enrichment procedure, preparative gel electrophoresis is a promising system: for knapsack instances with capacities corresponding to up to several tens of kilobases, the library to be searched can be narrowed down considerably. Exact solutions can then be obtained by sequencing of clones or microarray analysis.

The extension of the computation to include protein optimization is at the moment primarily of theoretical interest. The precise analysis of proteins is currently more technically challenging than DNA characterization. Also, formal knapsack problems are hard to encode in proteins, as the definitions ask for natural number values. The only natural number that proteins can provide is the number of amino acids they consist of, which is information already present in the encoding DNA. In contrast, the amino acid molecular weights used here are distributions over real numbers. Even so, the real interest of knapsack problems is not in formal definitions, but in physical occurrences, where natural numbers may or may not be applicable. One of these real world knapsack problems lies very close to the abstract computations implemented here: protein design. In this case, a complete protein is identified with a knapsack, and amino acids or protein domains with the items. The optimization function (in itself still poorly understood) of course considers far more complex properties than just molec-

ular weight, and the knapsack capacity is not as explicit as in the formal case. Still, limits do exist: proteins above a certain size may not pass cellular membrane pores, and infinitely long genes tend to be expressed at infinitely low rates. Nature's way of solving such problems has been the inspiration for *in silico* evolutionary algorithms, themselves often efficient methods to deal with some computationally hard problems. In this view, the molecular implementation of knapsack problems reported here can be considered a simplified case of directed protein evolution.

6

Summary and general discussion

Research on the computational application of DNA can be roughly divided into two categories. The first is an engineering discipline, where DNA and other biological macromolecules are recruited as components in actual computing devices. On the other hand, computing with DNA is helping to define new ways to think about computation. These two motivations do not necessarily coincide with experimental and theoretical investigations, respectively. Much theoretical effort is aimed at implementation issues such as sequence and algorithm design, and at the practical level, many experimental computations are not pursued beyond the 'proof of principle' stage of development.

The experiments described in this thesis fall mainly on the engineering side. The computations performed employ new DNA computer architectures and algorithms (chapters 2 and 3) or make use of technologies not incorporated in DNA computations before (chapters 3, 4 and 5). Notwithstanding this primary focus, some implementations might also hint at, or contribute to, new insights in the theory of molecular computing.

DNA computing with a single selection step

Chapters 2 and 3 pioneer the use of a recently described algorithm for molecular computing (Rozenberg & Spaink, 2003). The algorithm is based on DNA complementarity, and has the advantage of requiring only a single selection step: the separation of non-hybridized from hybridized DNA. Many molecular biological methods are available to accomplish this, however not all of them may be equally well tailored to molecular computing needs. Molecular methods concerned with biology usually deal with a few long molecule species of semi-random composition, whereas computing DNA is generally relatively short, of more ordered base sequence, and gathered in large libraries. In addition, most computing purposes require very strict (digital) signal discrimination, which DNA hybridization and subsequent duplex detection protocols do not naturally support.

The original proposal for the implementation of the blocking algorithm relies on PCR inhibition. By making certain duplexes (which do not correspond to solutions to a given problem) unavailable for DNA polymerase by means of modified nucleic acids, a PCR reaction on an initial pool of potential solutions should yield a mixture considerably enriched in species encoding proper answers. Initial experiments using peptide nucleic acid blockers (Ørum *et al.*, 1993) and 3' dideoxy modified oligonucleotides as PCR inhibiting species did not result in unambiguous discrimination between solution and non-solution molecules (unpublished results). Therefore, several alternative techniques were tested.

Chapter 2 describes the application of three assays for duplex detection, based on differential electrophoretic behaviour, susceptibility to mismatch endonuclease digestion, and fluorescence resonance energy transfer (FRET).

During electrophoresis, the mobility of homoduplex and heteroduplex DNA (perfectly matched or containing mismatched basepairs, respectively) differ, because distorted helices experience altered gel resistances. In principle, this effect can be used to distinguish oligonucleotides encoding satisfying assignments from falsified ones. While the former will not form a perfect duplex with a blocker and migrate at a predictable rate, the latter does.

Another method to discriminate between those classes is the utilization of mismatch endonucleases. Such enzymes recognize imperfectly matched DNA duplexes, and cut the phosphodiester backbone at mismatch positions. In theory, the result is not only the destruction of heteroduplex DNA, but also the formation of digestion fragments that give some information regarding the position of the mismatch. However, with the CEL I mismatch endonuclease employed in chapter 2, this information was effectively lost, as the enzyme also degraded homoduplex DNA to some extent.

Both heteroduplex recognizing methods were applied to a complete four variable, four clause 3SAT problem instance. This translates to 2^4 possible truth assignments, each encoded in an ssDNA library oligonucleotide, and eight possible falsifiers. Because of partly overlapping falsifying conditions for two clauses, only seven blocker oligonucleotides were used. The experimental computation then reduces to 112 hybridization evaluations. Using the heteroduplex migration assay, 82% of mismatched combinations could be detected, using CEL I this was 100%. In other words, the first method incorrectly classifies 19 permutations as falsified ones, in addition to the seven actual falsifying combinations. The mismatch endonuclease technique accurately detects only those seven. On the other hand, the heteroduplex assay has one major advantage over the CEL I assay: it is non-destructive, whereas the mismatch endonuclease actually preferentially digests the solutions to the problem instance. DNA classified by the gel migration technique can be salvaged by gel extraction methods, and perhaps be subjected to another iteration of selection using other constraints.

The last method tested in chapter 2, FRET, is likewise non-destructive. Other benefits of fluorescence methods in general include their speed and sensitivity. Also, these methods are routinely integrated with molecular biology, and many fluorescent dyes are available for covalent coupling to nucleic acids. FRET hybridization detection relies on the fact that when two different fluorophores are brought into close proximity, excitation energy can be transferred from a donor to an acceptor dye (depending on spectral characteristics of both). The latter then fluoresces, whereas in free solution only the former would exhibit fluorescence. The close proximity required is on the scale of DNA helix dimensions. In

chapter 2, the FRET technique was only applied to a few oligonucleotide pairs. Although the fluorescence signal allowed monitoring of the hybridization state, it was shown that it is difficult to encode sufficient information for a four variable SAT problem and remain within the spatial requirements imposed by FRET. With the algorithm and methods used, scaling to larger instances is probably incompatible with FRET detection.

In chapter 3, another fluorescence based method is evaluated for use in molecular computing. This technology, fluorescence correlation spectroscopy (FCS), is also capable of measuring DNA hybridization states. Its principle relies on altered Brownian motion of dsDNA compared to ssDNA in solution, resulting in longer diffusion times for the heavier dsDNA. Again, fluorescent labelling of the DNA oligomers is required. In a confocal detection volume, fluorescence signals are detected, and subsequently processed according to an auto-correlation function. The theoretical result is a shifted auto-correlation curve for samples containing dsDNA compared to those only containing ssDNA, which is due to the longer average time DNA duplexes spend in the detection volume. However, the signal difference between single-stranded and hybridized DNA was too low for reliable computation. Therefore, the method was extended to dual-colour FCS, in which both library and blocker strands are labelled with dyes with distinct excitation and emission spectra. Again, fluorescence signals are correlated, but this time in two separate channels. These correlation signals are then further processed to yield the cross-correlation signal, which contains information on the diffusional characteristics of particles containing both fluorescent dyes only (i.e. hybridized DNA). This method was applied to the entire set of hybridization evaluations mentioned above, and again reliable discrimination between falsified and solution assignments was achieved. The signal to noise ratio was the best of all methods tested for the blocking algorithm.

Successful computation is not the only accomplishment of the FCS DNA computing approach. The combination of high intensity laser excitation, minute (femtolitre) effective volumes and highly sensitive detectors enables a significant decrease in the detection limit for computing DNA compared to traditional output mechanisms, such as gel electrophoresis. Additional polymerase based signal amplification procedures, as employed in the majority of DNA computations reported to date (e.g. PCR and cloning, see chapter 1) also become superfluous. For practical reasons, the computation of chapter 3 was carried out using 10 µl samples: manual liquid handling and evaporation become major sources of error for lower volumes. With proper automation and microfluidic technology, however, volumes may be decreased dramatically, enabling the evaluation of literally single molecules.

Scaling issues

The computations of chapters 2 and 3 are on a four variable problem instance, which requires a total search space of 2^4 oligonucleotides. Every clause of the instance specifies a falsifying assignment, encoded in a blocking agent. However, since the formula contains only three literals per clause, the fourth variable remains unspecified in the blocker. In the described implementation, this resulted in two blocking species per clause, each with a different truth value for the fourth variable. If this strategy is pursued for larger instances (an n variable, m clause 3SAT instance), not only the number of possible assignments increases exponentially (2^n), but also the number of falsifiers ($m2^{n-3}$).

This is intrinsic to the nature of the computational problem and the DNA computing architecture. The blocking algorithm operates on global properties of molecules (i.e. hybridization), whereas others consider local properties (bit subsequences). The identity of every variable in an assignment is only set by its position in the corresponding molecule. While the blocker molecule actually only has to specify values for three variables, it lacks the possibility to specifically address only those variables. The linear nature of DNA then forces it to specify every variable physically placed in between the three under consideration. And since the possible sequences in between are exponential in number, so are the blocker molecules. This is a fundamental trade-off: algorithms that require only a single selection step (per clause) must address the entire assignment, and therefore lack the ability to specifically address variables. In contrast, an encoding according to Lipton (1995) allows one to target individual variables, owing to their unique sequences, but also requires distinct separation procedures for every literal.

Fortunately, there are several experimental escapes from this predicament. The easiest and cheapest is mixed base synthesis, in which instead of a single species of nucleotide, a specified mixture of nucleotides can be incorporated in a blocker oligonucleotide. In this way, an exponential variety of falsifying agents can be specified and synthesized in a linearly bounded number of steps. However, the resultant mixture will still contain $m2^{n-3}$ distinct species, which complicates experiments (see for example the multiple blocker additions in chapter 3). A more elegant solution is, to make just one species of molecule for every falsifier, in which the hybridization behaviour is indifferent to the complementary strand except at the specified variables. This can be achieved through the incorporation of abasic sites in the molecule (Kool, 1998) or so-called universal nucleotides (naturally occurring types such as deoxyinosine, Martin *et al.*, 1985, or artificial ones; Loakes, 2001). The number of blocking species then becomes equal to the number of clauses, m.

Another scaling issue is the use of dedicated gel lanes for every combination of blocker and assignment molecules, for example in the heteroduplex migration

experiment of chapter 2. While this does provide a lot of information on the efficiency of the selection procedure itself, it is ultimately undesirable because of the huge ($m2^n$) numbers of evaluations required. However, taking the experimental computation apart this way is not intrinsic to the method: in principle, a mixture of all library and all blocker molecules could be subjected to electrophoresis in a single lane. This does not immediately provide a detailed answer to the computation, but it does physically separate all satisfying from all non-satisfying assignments (thereby solving the equally hard associated decision problem). Similarly, all possible permutations may be included in a single mismatch endonuclease reaction, and the identity of surviving molecules can be obtained by alternative means. The FCS method is probably not that easily adjusted, as results from chapter 3 show a decrease in signal quality if samples are much more complex than consisting of two species. This may be a consequence of experimental conditions that can be further optimized, but still the method is essentially dependent on sequential evaluation of samples.

In vitro evolutionary algorithms for SAT

Regarding the questions from chapter 1, on the algorithms, methods and selection criteria for the implementation of evolutionary algorithms (EAs) in DNA, the results from chapters 2 and 3 and the discussion above provide the basis for the design outlined in figure 1a. Satisfiability problems are in principle open to evolutionary optimization, but care must be taken in the design of the algorithms (Gottlieb *et al.*, 2002). The most successful *in silico* approaches use a bit string representation, which is equivalent to the molecular bit string of the blocking algorithm. Variation is best achieved using the mutation operator, which is the flipping of individual bits. This is distinct from its biochemical counterpart: changing the value of the six nucleotide variables employed in this study is virtually impossible to achieve using random mutation. However, it is quite probable that the behaviour of DNA based evolutionary algorithms is very different from *in silico* counterparts, if only because the *in vitro* approach readily supports population sizes unattainable otherwise (typical values for EAs are of the order of one to 10 individuals).

DNA hybridization is a promising candidate for a selection criterion (as first proposed by Wood *et al.*, 1999). The methods described in this thesis may be implemented at different stages of the computation, for example the duplex migration assay of chapter 2 would make a good selection procedure. Key requirements for this stage in the computation are that it takes only a few steps, is highly parallel, and does not destroy molecules in the process. Somewhat problematically, the procedure is not entirely accurate, because of the lack of an absolute correlation between electrophoretic mobility and DNA complementa-

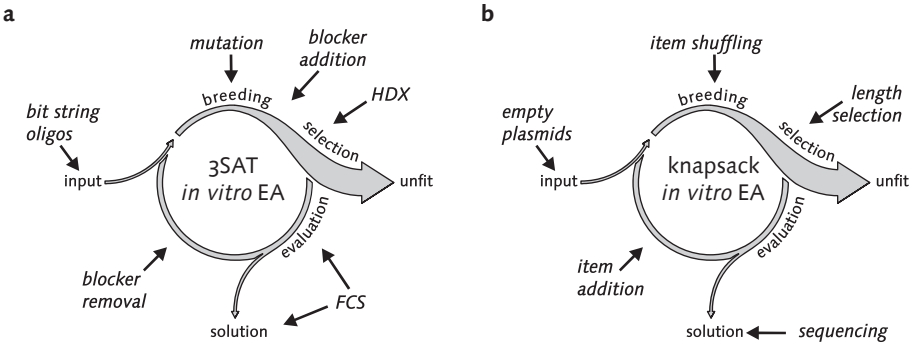

Figure 1. Suggestions for components of biomolecular evolutionary computations. **a** Satisfiability problems, based on experiments described in chapters 2 and 3. **b** Knapsack problems, based on chapter 5. Abbreviations: HDX, heteroduplex analysis; FCS, fluorescence correlation spectroscopy; EA, evolutionary algorithm. See text for further details.

rity (i.e. relatively highly mismatched ensembles may migrate closer to perfect duplexes than those with only a single mismatch). This can complicate the definition of a fitness function. However, for mismatch ratios of 5% and up, the migration behaviour does become proportional (Upchurch *et al.*, 2000). Also, the duplex migration assay is generally recommended for molecules of over 100 bp in length, as mismatch deformation is probably accentuated by longer duplex arms. Therefore, the precision of the selection step may even become higher for larger problem instances.

While electrophoretic behaviour may be used as an enrichment procedure, or a first relatively crude selection on huge numbers of candidate solutions, the superior discrimination and sensitivity of fluorescence correlation spectroscopy make the latter technique well suited for the subsequent examination of a limited number of candidate molecules. Further research for the implementation of this design should focus on parameter choice (mutation rates, population sizes, selection pressure), molecular breeding methods to achieve bit flipping, and finally integration of components.

Plasmid hardware

The experiments described in chapters 4 and 5 make use of a rather different approach than those discussed above. The computing architecture employed does not rely on single-stranded DNA, but instead uses plasmids, into which a special computational construct is introduced by cloning. This construct contains a bit string representation, which can be operated on by enzymes. Furthermore, the plasmid computing method depends on a 'mix-and-split' type of operation, in

which the entire assignment library can be split into several discrete volumes, followed by operations that are identical for all molecules in a volume (i.e. operations in parallel). It is assumed that because of diffusion and large numbers, every volume contains the same set of molecules before the operation. Afterwards, the now different volumes can be merged to form a new library. Recursive cycles of this type of operation can be employed to quickly generate a molecular library of exponential variety.

The nature of the bits varies with the implementation. In the computation by Head *et al.* (2000), bits are represented by 'stations' on the plasmid. These stations are flanked by unique restriction endonuclease recognition sites. Excision of a station followed by ligation of the plasmid changes the bit's value. Another strategy is chosen by Head *et al.* (2002a, b), where bit values are represented by the presence or absence of restriction sites: to change the value, a restriction enzyme cleaves its recognition site, which is subsequently modified by polymerase or exonuclease treatment. The plasmid is then religated, the net (irreversible) result being the removal of the restriction site. In both scenarios, the plasmid length is also changed slightly by bit operations, enabling selection of solution plasmids by gel electrophoresis. Since the parent plasmids used in chapters 4 and 5 are derivatives of the one used in Head *et al.* (2000), the experimental computations reported here apply bit modification of the first type.

Chapter 4 describes a computation to an instance of the Minimal Dominating Set (MDS) problem. The MDS problem is a graph problem, where every bit value is associated with the presence or absence of a node in a graph. The single, six station plasmid species at the initiation of the computation represents a graph with all six possible nodes included. The problem instance asks for the smallest possible set of nodes from which all other nodes in the graph can be reached by at most one connection. Using graph-specific information, a library of all possible solutions to the problem instance was generated by the mix-and-split methodology outlined above. After four iterations, a library of 16 different plasmid species was obtained. Owing to the length difference associated with bit values, identification and of the plasmid with the largest insert also solves the problem instance.

In previous DNA computations, and in particular in plasmid based computations, such an analysis always required DNA gel electrophoresis. The computation in chapter 4 introduces a new method for output, based on protein instead of nucleic acid. Since the computational construct is embedded in a hardware background amenable to general molecular biological technology, it is relatively easy to have it act in a more biological way: as a gene to direct the synthesis of a protein. For this purpose, the computational plasmid was provided with a strong inducible promoter and a sequence encoding a known protein epitope. These elements allow protein expression and purification, respectively, yielding a pro-

tein representation of the MDS library. Again, the experimental challenge is to identify the largest (heaviest) molecule, which relates to the smallest set of nodes asked for. This was achieved by MALDI-TOF spectroscopy on the protein mixture, which identified a largest protein with a mass matching the correct solution to the MDS instance.

An idealized DNA molecule allows for the storage of and computation with 2 bits per nucleotide. For computations completely relying on hybridization, the thermodynamics of duplex formation dictate constraints on the minimal bit sequence length. Hence, in chapters 2 and 3, DNA bit sequences were three or six nucleotides in length. With the plasmid used in chapter 4, the bit sequence length varies from 36 to 51 basepairs. While this is convenient for subsequent length discrimination by electrophoresis, the actual bit information can be stored in the (in this case) six basepairs required for restriction endonuclease recognition. This information redundancy is equally apparent in the protein representation, which contains at least four bits per amino acid. In other words, a single amino acid is in principle sufficient to represent a candidate solution to the MDS instance described above. This excess of molecular information inspired the computation described in chapter 5. Although stations are again binary bit representations, station length is also taken into account. The computation can be extended by protein expression, but in this case protein mass is not taken as a straightforward indicator of nucleic acid length but instead as an additional level of information. Consequently, stations contribute information through their presence or absence, length, and sequence.

The class of computational problems that fits these molecular parameters is that of knapsack problems. Given a volume of fixed capacity and a set of items of fixed size, knapsack problems ask for the subset of items that exactly fills the knapsack, or else approach its capacity as closely as possible. The optimization of knapsack contents can have more objectives than just volume: for example, items can be assigned a value, in which case the optimization criterion becomes maximum value in a limited volume. In chapter 5, computations to knapsack problems were implemented using the plasmid computing methodology by identifying stations with items. Their lengths correspond to item sizes; presence in the plasmid is interpreted as being included in the knapsack; mass of station translation products represents item value. On a seven item input, a library of 128 different knapsack contents was generated, from which plasmids exhibiting the correct size capacity were selected by gel electrophoresis. The possibility of value optimization was demonstrated, but the protein mixture was only superficially analysed using electrophoresis and not thoroughly characterized by mass spectrometry.

The outlook for plasmid and protein computing

The utilization of plasmids as components in DNA based computing has several advantages. Their circular nature enables the use of restriction endonucleases for bit modification. Without it, bit assignments would be cut in half for every modification, which complicates the ensuing ligation phase. In addition to keeping bits in context, the use of plasmids also serves a more practical goal: their superior handling in comparison to synthetic DNA. Not only do plasmids benefit from extensively optimized molecular protocols, they are also readily multiplied *in vivo*, which compares very favourably to *in vitro* methods in accuracy, reproducibility, cost and yield (PCR remains the method of choice for speed and selectivity). Plasmids are in general also of better and more consistent quality than synthetic sources of DNA, which lowers the potential for computational errors. Such qualities have recently enabled plasmids to also make their appearance in the DNA structural nanotechnology arena (Shih *et al.*, 2004).

If molecular computation is applied with the goal of obtaining a numerical output, the use of protein expression is perhaps somewhat premature. In theory, the deployment of computational protein libraries has many advantages: proteins have smaller mass, higher information content, and more programmable interactions than nucleic acids. However, in practice methods for the analysis of proteins are underdeveloped when compared to nucleic acid technology. This is for example apparent in the application of mass spectrometry as in chapter 4. Ultimately, the protein output to the computation was successful, but with considerably more experimental effort than straightforward DNA sequencing. Not only are protocols less standardized, but mass spectrometry detection of proteins also introduces a significant *in silico* computational overhead. Nevertheless, protein holds enormous computational potential, as does the combination of nucleic acids with protein computing (for example in multi-criterion optimization, chapter 5). Protein based computing requires further development of the understanding and prediction of interactions, and improved handling.

As in the case of the blocking algorithm, there may be some potential for the implementation of evolutionary algorithms in the plasmid computing strategy. Evolutionary computation has been successfully applied to knapsack problems *in silico* (see for example Laumanns *et al.*, 2004). Figure 1b outlines a recursive integer knapsack optimization algorithm. The integer knapsack problem, in contrast to the knapsack problems treated in chapter 5, supposes an unlimited supply of every item, i.e. there can be multiple (an integer) occurrences in the knapsack. As in chapter 5, item size and knapsack capacity are represented by DNA length. In figure 1b, the computation begins with an empty plasmid, containing no items. In every cycle an item can be added to all knapsacks, and in every cycle the knapsack contents are measured by gel electrophoresis. Selection criteria are

interactive: if all knapsacks are still below capacity, selection and item shuffling are not needed. However, as the knapsack grows to full capacity, only packings close to the knapsack threshold will be allowed to participate in the next cycle. This is also where item shuffling comes in: using recombination (if need be artificial, restriction enzyme mediated), new knapsack contents can evolve. Standard molecular length selection procedures will not accommodate an exact threshold in all cases: gel electrophoresis is limited to a resolution of about 0.1%. Therefore, for knapsack capacities over 1000, length selection will be an approximation, and final solutions will have to be determined by other methods (e.g. sequencing) from a pool enriched in knapsacks of roughly the correct size.

The evolution of molecular computation

Returning to the general subject of DNA computing as an engineering discipline, the question has risen whether DNA computations will ever be competitive, confronted with the enormous power of electronic computers. In the past ten years, no instances of NP-complete problems have been solved that were anywhere near threatening to even simple human trial and error efforts, no cryptography systems have been broken, and so on. It is hard to imagine how or why this should change in the near future (pending, of course, revolutionary new technologies). In fairness, computing with DNA is a spectacular invention (and discovery), but for now number-crunching is not its most likely application area.

The results presented in this thesis fit this general picture. Like in the work of others, advance is made in many aspects of molecular computation, but scaling to large instances remains elusive. Evolutionary computation seems promising to circumvent many implementation difficulties, and with considerable investments of time and effort the designs outlined in this chapter could work. They are certainly among the most realistic proposed for evolutionary DNA computing so far. Yet, from the point of view of computational power and efficiency, it is far from certain this time and effort will be well spent. If completed, these molecular evolutionary computers would still be single purpose, and of uncertain power. Programming would be extremely difficult, for example letting the integer knapsack EA run on a different input would require several weeks for the synthesis, cloning, and integrity checks of new DNA items. And a change of input is still far easier than one in purpose.

Then again, these molecular algorithms would be marvels of molecular control. The best recent illustration is the experimental computation by Braich *et al.* (2002), where a single species of DNA could be isolated from a pool of over a million other, but very similar ones. This is currently the pinnacle of precise detection in an ocean of molecular noise. Likewise, chapter 2 and 3 show high

fidelity discrimination among over a hundred very similar species. Such developments are themselves already illustrations of the value of DNA computing, for such methods may be useful for diagnostic or analytical purposes outside of computation. Other spin-offs, i.e. technologies to which experimental DNA computing contributes, include DNA surface chemistry, *in vitro* evolution and nanotechnology.

An emerging trend in DNA computing research is not to be concerned with computational power, but instead focus on the proven qualities of nucleic acids: they are small, and they are biologically significant. Small computers may simply be used to perform computations, of arbitrary power and complexity, in inaccessible places. The significance of nucleic acids is evident, and the combination of both qualities quickly results in speculative designs for nanoscale computers that perform computations on biological input, and yield a biological output. Such devices could find wide employ in biotechnology, from environmental to medical. An interesting application is a form of smart gene therapy, in which minute computers may diagnose single cells and administer molecular genetic treatment (for example an RNA species). Perhaps such computers will take the form of genetically engineered bacteria, with simple inbuilt logic 'circuitry' (Kobayashi *et al.*, 2004).

DNA computing as an engineering discipline is therefore very much alive, and already expanding beyond what is traditionally seen as computation. The other purpose of the field is theoretical: to help shape new conceptual frameworks for computing. Even experiments in DNA computing can be demonstrations of such new concepts. As an example, in the computations in chapters 2 and 3 the traditional distinction between hardware and software has vanished – both roles are assumed by single-stranded DNA.

Another way to emphasize this concept is to consider the inherently physical nature of DNA computers. Information and computation become much more tangible than they are in electronic processes. The most interesting aspect of this physicality may be the essentially liquid nature of molecular computers. The DNA computing architecture that is most explicitly concerned with this feature is aqueous computing (Head, 2000; Head *et al.*, 2002b), of which the plasmid computing experiments described in chapters 4 and 5 are manifestations. It uses the fundamental properties of molecular computers mentioned in chapter 1 (three-dimensionality, free movement) to define a fluid computer memory without register, in which components lack an address, and are operated on by a combination of physical separation and Brownian search principles.

Finally, computing with DNA contributes to biological thought as well, through the identification of processes of life with computation. As a complement to this theoretical approach to natural computing, experimental DNA based computing may be seen as a manifestation of what is sometimes called

'synthetic biology' (Benner, 2003): in this case to understand the computational qualities of biology through implementing computations using biological macromolecules. An example of such computational synthetic biology is the analogy between protein design and knapsack optimization, suggested in chapter 5.

Samenvatting

Er zijn verschillende redenen om moleculaire computers te willen maken. Een vaak genoemde is, dat de huidige tendens van miniaturisering in de chiplithografie (ook bekend als de Wet van Moore) over enkele tientallen jaren onherroepelijk zal leiden tot computercomponenten die dicht tegen de moleculaire schaal aan zitten. Het is onwaarschijnlijk dat dergelijke computers nog volgens dezelfde vertrouwde beginselen kunnen werken (het doorgeven van elektronen), dus wordt er nu al onderzoek gedaan naar alternatieven. Ondanks de haken en ogen aan deze overweging, staat het wel vast dat moleculaire computers veel interessante mogelijkheden hebben.

Een van de moleculen die de laatste jaren is onderzocht op rekenkundig vermogen is DNA, in het dagelijks leven de drager van erfelijke informatie. Juist vanwege die natuurlijke rol zijn op DNA gebaseerde computers voor moleculair biologen niets nieuws. Processen in de cel hebben alles te maken met informatieoverdracht en -bewerking, en een cel kan dus gezien worden als een complex stelsel van moleculaire berekeningen. Wat wel nieuw is, is het gebruik van biologische moleculen in kunstmatige computers.

DNA lijkt de ideale kandidaat voor kunstmatige moleculaire computers. De manier waarop het informatie bevat is niet al te exotisch: de volgorde van de basen A, T, C, en G is maar iets ingewikkelder dan de voor elektronische computers gebruikelijke binaire notatie. Daarnaast is er de afgelopen decennia een enorme gereedschapskist aan moleculair biologische technieken ontwikkeld om de informatie in DNA te bewerken. Ook de voorspelbare interacties van DNA zijn van belang: door de complementariteit van de nucleotiden (de baseparen A–T en C–G) is de binding tussen moleculen te programmeren. Complementaire stukjes enkelstrengs DNA in oplossing 'hybridiseren' spontaan tot dubbelstrengs DNA (met de bekende dubbele-helixstructuur). Andere moleculen, zoals eiwitten, zijn in beginsel ook geschikt, maar hun structuren en interacties zijn voorlopig nog te onvoorspelbaar om goed programmeerbare computers mogelijk te maken.

In 1994 werd de eerste berekening met DNA uitgevoerd. Met behulp van enkele oligonucleotiden (korte stukjes DNA) in oplossing werd een antwoord gevonden voor een klein voorbeeld van een ingewikkeld wiskundig vraagstuk, verwant aan het handelsreizigersprobleem. Dit experiment zette de trend voor de meeste volgende berekeningen, die eveneens oplossingen geven voor zogenaamde NP volledige problemen. Deze belangrijke categorie problemen wordt gekenmerkt door een zeer groot aantal mogelijke oplossingen (exponentieel in verhouding tot de grootte van de invoer), die eigenlijk allemaal geverifieerd moeten worden. Voor problemen van relevante grootte is dit onbegonnen werk, ook met behulp van een elektronische computer: alle oplossingen moeten één voor één (sequentieel) worden bekeken. Met behulp van DNA kan dat anders: alle oplossingen kunnen tegelijkertijd gemaakt en geëvalueerd worden (parallel), dankzij de vele miljarden moleculen in oplossing die interacties met elkaar aangaan.

Er zijn tot nu toe ongeveer 15 studies gepubliceerd waarin daadwerkelijk werd gerekend met DNA. Alle experimenten gebruiken een andere combinatie van algoritmen en technieken, en het is nog niet duidelijk wat nu de effectiefste strategie is (hoofdstuk 1 geeft een samenvatting van de pogingen). Wat wel duidelijk naar voren komt, is de moeilijkheid van het opschalen van de berekeningen. Het blijkt lastig de benodigde grote verzamelingen DNA moleculen aan te maken en te bewerken. De belangrijkste reden hiervoor is, dat de huidige DNA technologie toch te beperkt is. De methoden zijn eenvoudigweg te grof: in de moleculaire biologie heeft de nadruk nooit gelegen op de ontwikkeling van 'digitale' scheidings- en analysetechnieken. De onvermijdelijke ruis in de experimentele uitvoering beperkt de maximale grootte van de mogelijke input en verstoort het verloop van langdurige berekeningen.

Een mogelijke, en heel natuurlijke uitweg is evolutionaire optimalisatie van kandidaat oplossingen. In de informatica staan zulke methoden bekend als evolutionaire algoritmen. De moleculaire biologie kan de technieken leveren voor *in vitro* (in plaats van *in silico*) evolutie van kandidaat oplossingen. Zulke gestuurde evolutie door natuurlijke selectie wordt al gebruikt voor het ontwikkelen van biologisch actieve stoffen, waaronder ook eiwitten en DNA. Het grootste voordeel van de combinatie van DNA computing en evolutionaire algoritmen zou de enorme oplossingsruimte zijn die doorzocht kan worden. Het is echter niet duidelijk of dit haalbaar is, en welke algoritmen en technieken een rol zouden kunnen spelen. In twee verschillende onderzoekslijnen zijn enkele veelbelovende mogelijkheden onderzocht.

De eerste mogelijkheid is gebaseerd op een recent beschreven moleculair algoritme. Het voordeel van dit algoritme is, dat er maar één scheidingsstap nodig is om het DNA dat de oplossing van een probleem codeert te isoleren. Alle andere werkwijzen hebben, afhankelijk van de grootte van het bestudeerde vraagstuk, vele stappen nodig. Door het aantal scheidingsstappen klein te houden, zou het uitvoeren van een evolutionaire cyclus (waarin deze scheidingsstap keer op keer moet worden uitgevoerd) veel eenvoudiger kunnen worden.

De scheiding die nodig is, is tussen enkelstrengs en dubbelstrengs (complementair gehybridiseerd) DNA, waarbij de eerste categorie de juiste oplossingen geeft. Er bestaan verschillende moleculair biologische strategieën die hiervoor geschikt zouden kunnen zijn. Echter, alle protocollen zijn ontworpen en geoptimaliseerd met biologische doelen en moleculen in gedachten, en het is dus helemaal niet zeker of zij aan de strenge eisen van DNA computing voldoen. De eerste poging, met behulp van PCR (polymerase kettingreactie, een selectieve vermeerderingstechniek), gaf geen duidelijk resultaat. Daarna werd hybridisatie detectie met behulp van FRET (een fluorescentie techniek) geprobeerd. Hoewel dit werkt, werd ook duidelijk dat deze methode niet geschikt is voor grotere vraagstukken. Twee andere technieken, die normaal worden gebruikt voor het

opsporen van mutaties in biologisch DNA, waren wel redelijk succesvol (hoofdstuk 2). Beiden werden getoetst op een logisch *Satisfiability* vraagstuk met 4 variabelen.

Ditzelfde vraagstuk is ook gebruikt voor het testen van een *single-molecule* spectroscopie techniek. Eén van de knelpunten bij het uitvoeren van moleculaire berekeningen is de output. Er is altijd meer DNA nodig geweest als waarneembaar antwoord dan voor de berekeningen zelf. Meestal wordt voor het uitlezen van het resultaat gel-electroforese gebruikt, waar vele miljarden moleculen voor nodig zijn. Bij de berekening beschreven in hoofdstuk 3 werden de output moleculen geanalyseerd met behulp van een geavanceerde techniek, fluorescentie correlatie spectroscopie (FCS). Deze methode maakt het mogelijk bepaalde eigenschappen van moleculen in oplossing te achterhalen. Vanwege het extreem kleine detectievolume (ongeveer 1 kubieke micrometer, een femtoliter) en de zeer hoge gevoeligheid kunnen deze metingen in theorie aan enkele moleculen plaatsvinden. Nu is het erg onpraktisch om werkelijk met slechts een paar moleculen in oplossing te werken, maar het bleek wel mogelijk de benodigde hoeveelheid DNA voor een berekening drastisch te verlagen, en toch een betrouwbaar antwoord te krijgen. De bezwaren voor het nog verder reduceren van de hoeveelheid DNA zijn vooral gelegen in de moeilijkheid van het werken met minieme volumes water (oplosmiddel): zo verdampt een femtoliter gegarandeerd voor er aan kan worden gemeten. Het moet echter mogelijk zijn de FCS metingen te combineren met recent ontwikkelde *lab-on-a-chip*-technologieën om de output werkelijk terug te brengen tot enkele moleculen.

Voor grotere problemen (en evolutionaire optimalisatie) lijkt het gebruik van een combinatie van technieken veelbelovend: eerst een relatief grove scheiding van veel kandidaat oplossingen met behulp van een mutatie detectie methode, gevolgd door een zeer nauwkeurige single-molecule aanpak.

Een andere mogelijkheid die is onderzocht is heel anders van opzet. De selectie is hier niet op de complementariteit van DNA, maar op de lengte van het polymeer. Ook deze selectie is vrij snel en eenvoudig uit te voeren, bijvoorbeeld met behulp van gel-electroforese. De methode is met succes getest op verschillende wiskundige vraagstukken. Het gebruikte DNA was zodanig ontworpen, dat het ook *in vivo* kon worden vertaald in een eiwit. Het bleek ook mogelijk antwoorden te bepalen op eiwitniveau, na karakterisering met behulp van massaspectrometrie (hoofdstuk 4). De uitbreiding van moleculair rekenen met eiwitten is interessant, bijvoorbeeld omdat zo verschillende eigenschappen tegelijkertijd kunnen worden geoptimaliseerd (hoofdstuk 5). Bovendien biedt het wellicht perspectieven voor evolutionaire optimalisatie, want biologische evolutie grijpt ook aan op eigenschappen van eiwitten.

De toekomst van DNA berekeningen voor zeer complexe vraagstukken ligt zeker in de evolutionaire aanpak. Ook geavanceerde technieken als de ontwikkel-

de FCS output methode zouden hierin kunnen worden ingepast. Maar voorlopig kunnen DNA computers helemaal nog niet concurreren met hun elektronische neefjes, en het is maar de vraag of dit ooit wel het geval zal zijn. Er zijn echter nog veel andere toepassingen denkbaar voor DNA computers. Zo hoeven echt niet alle computers extreem krachtig te zijn, er zijn ook niches voor zeer kleine computers, zeer energiezuinige computers, en computers met een biologische interface. Ook voor al deze toepassingen zijn de ontwikkelde technieken van belang. Tegenwoordig worden de berekeningen aan wiskundige problemen zelfs gezien als 'benchmarks' voor nieuwe technologie. En dat heeft ook weer voordelen voor de moleculaire biologie: DNA computing blijkt vaak een katalysator voor de ontwikkeling van nieuwe, gevoeliger en meer betrouwbare methoden.

References

L.M. Adleman (1994) Molecular computation of solutions to combinatorial problems. *Science* **266**, 1021–1024

L.M. Adleman (1996) On constructing a molecular computer. In: R.J. Lipton & E. Baum (eds.) *DNA based computers, DIMACS 27*. American Mathematical Society, Providence, RI, pp 1–21

L.M. Adleman (1998) Computing with DNA. *Sci. Am.* **279**, 54–61

Y. Aoi, T. Yoshinobu, K. Tanizawa, K. Kinoshita & H. Iwasaki (1998) Solution of the knapsack problem by deoxyribonucleic acid computing. *Jpn. J. Appl. Phys. Part 1* **37**, 5839–5841

N.P. Armitage, M. Briman & G. Grüner (2004) Charge transfer and charge transport on the double helix. *Phys. Stat. Sol. B* **241**, 69–75

A. Bachtold, P. Hadley, T. Nakanishi & C. Dekker (2001) Logic circuits with carbon nanotube transistors. *Science* **294**, 1317–1320

K. Bacia, I.V. Majoul & P. Schwille (2002) Probing the endocytic pathway in live cells using dual-color fluorescence cross-correlation analysis. *Biophys. J.* **83**, 1184–1193

C. Bancroft, T. Bowler, B. Bloom & C.T. Clelland (2001) Long-term storage of information in DNA. *Science* **293**, 1763–1765

R. Bar-Ziv, T. Tlusty & A. Libchaber (2002) Protein-DNA computation by stochastic assembly cascade. *Proc. Natl. Acad. Sci. USA* **99**, 11589–11592

E.B. Baum (1995) Building an associative memory vastly larger than the brain. *Science* **268**, 583–585

T. Bäck, J.N. Kok & G. Rozenberg (2003) Evolutionary computation as a paradigm for DNA-based computing. In: L.F. Landweber & E. Winfree (eds.) *Evolution as Computation. DIMACS workshop, Princeton, January 1999*. Springer-Verlag, Berlin Heidelberg, pp 15–40

Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh & E. Shapiro (2001) Programmable and autonomous computing machine made of biomolecules. *Nature* **414**, 430–434

Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh & E. Shapiro (2003) DNA molecule provides a computing machine with both data and fuel. *Proc. Natl. Acad. Sci. USA* **100**, 2191–2196

Y. Benenson, B. Gil, U. Ben-Dor, R. Adar & E. Shapiro (2004) An autonomous molecular computer for logical control of gene expression. *Nature* **429**, 423–429

S.A. Benner (2003) Act natural. *Nature* **421**, 118

C.H. Bennett (1973) Logical reversibility of computation. *IBM J. Res. & Dev.* **17**, 525–532

C.H. Bennett (1982) The thermodynamics of computation – a review. *Int. J. Theor. Phys.* **21**, 905–940

C.H. Bennett & R. Landauer (1985) The fundamental physical limits of computation. *Sci. Am.* **253**, 48–56

C.H. Bennett & D.P. DiVincenzo (2000) Quantum information and computation. *Nature* **404**, 247–255

S. Bernacchi & Y. Mély (2001) Exciton interaction in molecular beacons: a sensitive sensor for short range modifications of the nucleic acid structure. *Nucleic Acids Res.* **29**, e62

S. Bernacchi, E. Piémont, N. Potier, A. van Dorsselaer & Y. Mély (2003) Excitonic heterodimer formation in an HIV-1 oligonucleotide labeled with a donor-acceptor pair used for fluorescence resonance energy transfer. *Biophys. J.* **84**, 643–654

V. Bhalla, R.P. Bajpai & L.M. Bharadwaj (2003) DNA electronics. *EMBO Rep.* **4**, 442–445

R.R. Birge, N.B. Gillespie, E.W. Izaquirre, A. Kusnetzow, A.F. Lawrence, D. Singh, W. Song, E. Schmidt, J.A. Stuart, S. Seetharaman & K.J. Wise (1999) Biomolecular electronics: protein-based associative processors and volumetric memories. *J. Phys. Chem. B* **103**, 10746–10766

P.S. Blank, C.M. Sjomeling, P.S. Backlund & A.L. Yergey (2002) Use of cumulative distribution functions to characterize mass spectra of intact proteins. *J. Am. Soc. Mass Spectrom.* **13**, 40–46

R.S. Braich, C. Johnson, P.W.K. Rothemund, D. Hwang, N. Chelyapov & L.M. Adleman (2001) Solution of a satisfiability problem on a gel-based DNA computer. In: A. Condon & G. Rozenberg (eds.) *DNA computing, 6th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 27-42

R.S. Braich, N. Chelyapov, C. Johnson, P.W.K. Rothemund & L. Adleman (2002) Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* **296**, 499–502

E. Braun & K. Keren (2004) From DNA to transistors. *Adv. Phys.* **53**, 441–496

D. Bray (1995) Protein molecules as computational elements in living cells. *Nature* **376**, 307–312

R.R. Breaker (2000) Making catalytic DNAs. *Science* **290**, 2095–2096

R.R. Breaker (2002) Engineered allosteric ribozymes as biosensor components. *Curr. Opin. Biotech.* **13**, 31–39

A. Brenneman & A. Condon (2002) Strand design for biomolecular computation. *Theor. Comput. Sci.* **287**, 39–58

S. Brenner, S.R. Williams, E.H. Vermaas, T. Storck, K. Moon, C. McCollum, J.-I. Mao, S. Luo, J.J. Kirchner, S. Eletr, R.B. DuBridge, T. Burcham & G. Albrecht (2000) *In vitro* cloning of complex mixtures of DNA on microbeads: physical separation of differentially expressed cDNAs. *Proc. Natl. Acad. Sci. USA* **97**, 1665–1670

J. Brown, T. Brown & K.R. Fox (2001) Affinity of mismatch-binding protein MutS for heteroduplexes containing different mismatches. *Biochem. J.* **354**, 627–633

C.T. Bui, K. Rees, A. Lambrinakos, A. Bedir & R.G.H. Cotton (2002) Site-selective reactions of imperfectly matched DNA with small chemical molecules: applications in mutation detection. *Bioorg. Chem.* **30**, 216–232

B. Bunow (1995) Letters: on the potential of molecular computing. *Science* **268**, 482–483

A.W. Burks, H.H. Goldstine & J. von Neumann (1946) Preliminary discussion of the logical design of an electronic computing instrument. In: A.H. Taub (ed.) *Collected works of John von Neumann, vol. 5 (1963)*. The Macmillan company, New York, pp 34–79

R.A. Cardullo, S. Agrawal, C. Flores, P.C. Zamecnik & D.E. Wolf (1988) Detection of nucleic acid hybridization by nonradiative fluorescence resonance energy transfer. *Proc. Natl. Acad. Sci. USA* **85**, 8790–8794

M.J. Chalmers & S.J. Gaskell (2000) Advances in mass spectrometry for proteome analysis. *Curr. Opin. Biotech.* **11**, 384–390

W.-L. Chang, M.S.-H. Ho & M. Guo (2004) Molecular solutions for the subset-sum problem on DNA-based supercomputing. *Biosystems* **73**, 117–130

J. Chen, R. Deaton, M. Garzon, J.W. Kim, D. Wood, H. Bi, D. Carpenter & Y.Z. Wang (2004) Characterization of non-crosshybridizing DNA oligonucleotides manufactured *in vitro*. In: C. Ferretti, G. Mauri & C. Zandron (eds.) *DNA10, tenth international meeting on DNA computing, preliminary proceedings*. Università di Milano-Bicocca, pp 132–141

J.H. Chen & D.H. Wood (2000) Computation with biomolecules. *Proc. Natl. Acad. Sci. USA* **97**, 1328–1330

D.T. Chiu, E. Pezzoli, H. Wu, A.D. Stroock & G.M. Whitesides (2001) Using three-dimensional microfluidic networks for solving computationally hard problems. *Proc. Natl. Acad. Sci. USA* **98**, 2961–2966

A. Chworos, I. Severcan, A.Y. Koyfman, P. Weinkam, E. Oroudjev, H.G. Hansma & L. Jaeger (2004) Building programmable jigsaw puzzles with RNA. *Science* **306**, 2068–2072

C.T. Clelland, V. Risca & C. Bancroft (1999) Hiding messages in DNA microdots. *Nature* **399**, 533–534

M. Conrad (1985) On design principles for a molecular computer. *Commun. ACM* **28**, 464–480

M. Conrad (1992) Molecular computing paradigms. *Computer* **25**, 6–9

J.P.L. Cox (2001) Long-term data storage in DNA. *Trends Biotechnol.* **19**, 247–250

R. Deaton, R.C. Murphy, J.A. Rose, M. Garzon, D.R. Franceschetti & S.E. Stevens Jr. (1997) A DNA based implementation of an evolutionary search for good encodings for DNA computation. *Proceedings of the fourth IEEE conference on evolutionary computation, Indianapolis, IN*. IEEE Press, Piscataway, NJ, pp 267–271

R. Deaton, J. Chen, H. Bi, M. Garzon, H. Rubin & D.H. Wood (2003) A PCR-based protocol for *in vitro* selection of non-crosshybridizing oligonucleotides. In: M. Hagiya & A. Ohuchi (eds.) *DNA computing, 8th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 196–204

R.M. Dirks, M. Lin, E. Winfree & N.A. Pierce (2004) Paradigms for computational nucleic acid design. *Nucleic Acids Res.* **32**, 1392–1403

K.E. Drexler (1981) Molecular engineering: an approach to the development of general capablities for molecular manipulation. *Proc. Natl. Acad. Sci. USA* **78**, 5275–5278

A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott & G. Rozenberg (2004) Computation in living cells – gene assembly in ciliates. Springer-Verlag, Heidelberg Berlin

A.E. Eiben & J.E. Smith (2003) *Introduction to evolutionary computing*. Springer-Verlag, Berlin Heidelberg

M. Eigen & R. Rigler (1994) Sorting single molecules: application to diagnostics and evolutionary biotechnology. *Proc. Natl. Acad. Sci. USA* **91**, 5740–5747

D. Faulhammer, A.R. Cukras, R.J. Lipton & L.F. Landweber (2000) Molecular computation: RNA solutions to chess problems. *Proc. Natl. Acad. Sci. USA* **97**, 1385–1389

U. Feldkamp, H. Rauhe & W. Banzhaf (2003) Software tools for DNA sequence design. *Genetic Programming and Evolvable Machines* **4**, 153–171

R.P. Feynman (1959) There's plenty of room at the bottom. Reprinted in: *Journal of Microelectromechanical Systems* (1992) **1**, 60–66

A.S. Fraenkel (1999) Protein folding, spin glass and computational complexity. In: H. Rubin & D.H. Wood (eds.) *DNA based computers III, proceedings DIMACS workshop*. American Mathematical Society, Providence, RI, pp 101–121

A. Ganguly, M.J. Rock & D.J. Prockop (1993) Conformation-sensitive gel electrophoresis for rapid detection of single-base differences in double-stranded PCR products and DNA fragments: evidence for solvent-induced bends in DNA heteroduplexes. *Proc. Natl. Acad. Sci. USA* **90**, 10325–10329

T.S. Gardner, C.R. Cantor & J.J. Collins (2000) Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**, 339–342

M.R. Garey & D.S. Johnson (1979) *Computers and intractability. A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York

A. Gehani & J. Reif (1999) Micro flow bio-molecular computation. *Biosystems* **52**, 197–216

D.K. Gifford (1994) On the path to computation with DNA. *Science* **266**, 993–994

E. Goode, D.H. Wood & J. Chen (2001) DNA implementation of a royal road fitness evaluation. In: A. Condon & G. Rozenberg (eds.) *DNA Computing, proceedings 6th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 247–262

J. Gottlieb, E. Marchiori & C. Rossi (2002) Evolutionary algorithms for the Satisfiability problem. *Evol. Comput.* **10**, 35–50

F. Guarnieri, M. Fliss & C. Bancroft (1996) Making DNA add. *Science* **273**, 220–223

M. Hagiya (2001) From molecular computing to molecular programming. In: A. Condon & G. Rozenberg (eds.) *DNA computing, 6th international workshop on DNA-based computers*. Springer Verlag, Berlin Heidelberg, pp 89–102

D.R. Halpin & P.B. Harbury (2004) DNA display II. Genetic manipulation of combinatorial chemistry libraries for small-molecule evolution. *PLoS Biol.* **2**, e174

J. Hartmanis (1995) On the weight of computations. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* **55**, 136–138

J. Hasty, D. McMillen & J.J. Collins (2002) Engineered gene circuits. *Nature* **420**, 224–230

T. Head (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *B. Math. Biol.* **49**, 737–759

T. Head (2000) Circular suggestions for DNA computing. In: A. Carbone, M. Gromov & P. Prusinkiewcz (eds.) *Pattern formation in biology, vision and dynamics*. World Scientific, Singapore, pp 325–335

T. Head, G. Rozenberg, R.S. Bladergroen, C.K.D. Breek, P.H.M. Lommerse & H.P. Spaink (2000) Computing with DNA by operating on plasmids. *Biosystems* **57**, 87–93

T. Head, X. Chen, M. Yamamura & S. Gal (2002a) Aqueous computing: a survey with an invitation to participate. *J. Comput. Sci. Technol.* **17**, 672–681

T. Head, X. Chen, M.J. Nichols & S. Gal (2002b) Aqueous solutions of algorithmic problems: emphasizing knights on a 3x3. In: N. Jonoska & N.C. Seeman (eds.) *DNA computing, 7th international meeting on DNA based computers*. Springer-Verlag, Berlin Heidelberg, pp 191–202

P.N. Hengen, I.G. Lyakhov, L.E. Stewart & T.D. Schneider (2003) Molecular flip-flops formed by overlapping Fis sites. *Nucleic Acids Res.* **31**, 6663–6673

W.E. Highsmith, Q. Jin, A.J. Nataraj, J.M. O'Connor, V.D. Burland, W.R. Baubonis, F.P. Curtis, N. Kusukawa & M.M. Garner (1999) Use of a DNA toolbox for the characterization of mutation scanning methods. I: Construction of the toolbox and evaluation of heteroduplex analysis. *Electrophoresis* **20**, 1186–1194

A. Hjelmfelt & J.Ross (1995) Implementation of logic functions and computations by chemical kinetics. *Physica D* **84**, 180–193

H. Hug & R. Schuler (2002) DNA-based parallel computating of simple arithmetic. In: N. Jonoska & N.C. Seeman (eds.) *DNA computing, 7th international meeting on DNA based computers*. Springer-Verlag, Berlin Heidelberg, pp 321–328

International Technology Roadmap for Semiconductors (2003 edition) *Executive Summary*. http://public.itrs.net/files/2003itrs/home2003.htm

Intel Corporation (2004) *Microprocessor quick reference guide*. http://www.intel.com/pressroom/kits/quickref.htm

G.F. Joyce (2004) Directed evolution of nucleic acid enzymes. *Annu. Rev. Biochem.* **73**, 791–836

P.D. Kaplan, Q. Ouyang, D.S. Thaler & A. Libchaber (1997) Parallel overlap assembly for the construction of computational DNA libraries. *J. Theor. Biol.* **188**, 333–341

K. Keren, R.S. Berman, E. Buchstab, U. Sivan & E. Braun (2003) DNA-templated carbon nanotube field-effect transistor. *Science* **302**, 1380–1382

D. Kim, S.-Y. Shin, I.-H. Lee & B.-T. Zhang (2003) NACST/Seq: a sequence design system with multiobjective optimization. In: M. Hagiya & A. Ohuchi (eds.) *DNA computing, 8th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 242–251

H. Kobayashi, M. Kærn, M. Araki, K. Chung, T.S. Gardner, C.R. Cantor & J.J. Collins (2004) Programmable cells: interfacing natural and engineered gene networks. *Proc. Natl. Acad. Sci. USA* **101**, 8414–8419

K. Komiya, K. Sakamoto, H. Gouzu, S. Yokoyama, M. Arita, A. Nishikawa & M. Hagiya (2001) Successive state transitions with I/O interface by molecules. In: A. Condon & G. Rozenberg (eds.) *DNA computing, 6th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 17–26

E.T. Kool (1998) Replication of non-hydrogen bonded bases by DNA polymerases: a mechanism for steric matching. *Biopolymers* **48**, 3–17

K. Korn, P. Gardellin, B. Liao, M. Amacker, Å. Bergström, H. Björkman, A. Camacho, S. Dörhöfer, K. Dörre, J. Enström, T. Ericson, T. Favez, M. Gösch, A. Honegger, S. Jaccoud, M. Lapczyna, E. Litborn, P. Thyberg, H. Winter & R. Rigler (2003) Gene expression analysis using single molecule detection. *Nucleic Acids Res.* **31**, e89

V.N. Kristensen, D. Kelefiotis, T. Kristensen & A.L. Borresen-Dale (2001) High-throughput methods for detection of genetic variation. *Biotechniques* **30**, 318–332

J.R. Lakowicz (1999) *Principles of fluorescence spectroscopy, second edition*. Kluwer Academic / Plenum Publishers, New York

L.F. Landweber, T.-C. Kuo & E.A. Curtis (2000) Evolution and assembly of an extremely scrambled gene. *Proc. Natl. Acad. Sci. USA* **97**, 3298–3303

M. Laumanns, L. Thiele & E. Zitzler (2004) Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Nat. Comput.* **3**, 37–51

C.-M. Lee, S.W. Kim, S.M. Kim & U. Sohn (1999) DNA computing the Hamiltonian Path Problem. *Mol. Cells* **9**, 464–469

J.Y. Lee, S.-Y. Shin, S.J. Augh, T.H. Park & B.-T. Zhang (2003) Temperature gradient-based DNA computing for graph problems with weighted edges. In: M. Hagiya & A. Ohuchi (eds.) *DNA computing, 8th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 73–84

J.Y. Lee, S.-Y. Shin, T.H. Park & B.-T. Zhang (2004) Solving travelling salesman problems with DNA molecules encoding numerical values. *Biosystems* **78**, 39–47

A. Leier, C. Richter, W. Banzhaf & H. Rauhe (2000) Cryptography with DNA binary strands. *Biosystems* **57**, 13–22

Y. Li & R.R. Breaker (1999) Kinetics of RNA degradation by specific base catalysis of transesterification involving the 2'-hydroxyl group. *J. Am. Chem. Soc.* **121**, 5364–5372

T. Lindahl (1993) Instability and decay of the primary structure of DNA. *Nature* **362**, 709–715

R.J. Lipton (1995) DNA solution of hard computational problems. *Science* **268**, 542–545

Q. Liu, L. Wang, A.G. Frutos, A.E. Condon, R.M. Corn & L.M. Smith (2000) DNA computing on surfaces. *Nature* **403**, 175–179

Y. Liu, J. Xu, L. Pan & S. Wang (2002) DNA solution of a graph coloring problem. *J.Chem. Inf. Comput. Sci.* 42, 524–528

D. Liu, S.H. Park, J.H. Reif & T.H. LaBean (2004) DNA nanotubes self-assembled from triple-crossover tiles as templates for conductive nanowires. *Proc. Natl. Acad. Sci. USA* **101**, 717–722

D. Loakes (2001) The applications of universal DNA base analogues. *Nucleic Acids Res.* **29**, 2437–2447

M. Lundstrom (2003) Moore's law forever? *Science* **299**, 210–211

D.A. Mac Dónaill (1996) On the scalability of molecular computational solutions to NP problems. *J. Univers. Comput. Sci.* **2**, 87–95

B.J. MacLennan (2003) Transcending Turing computability. *Minds Machines* **13**, 3–22

D. Madge, E. Elson & W.W. Webb (1972) Thermodynamic fluctuations in a reacting system – measurement by fluorescence correlation spectroscopy. *Phys. Rev. Lett.* **29** , 705–708

C.D. Mao, T.H. LaBean, J.H. Reif & N.C. Seeman (2000) Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature* **407**, 493–496

S.A.E. Marras, F.R. Kramer & S. Tyagi (2002) Efficiencies of fluorescence resonance energy transfer and contact-mediated quenching in oligonucleotide probes. *Nucleic Acids Res.* **30**, e122

F.H. Martin, M.M. Castro, F. Aboul-ela & I. Tinoco Jr. (1985) Base pairing involving deoxyinosine: implications for probe design. *Nucleic Acids Res.* **13**, 8927–8938

R.D. Mashal, J. Koontz & J. Sklar (1995) Detection of mutations by cleavage of DNA heteroduplexes with bacteriophage resolvases. *Nature Genet.* **9**, 177–183

G. Mauri & C. Ferretti (2004) Word design for molecular computing: a survey. In: J. Chen & J. Reif (eds.) *DNA computing, 9th international workshop on DNA based computers*. Springer-Verlag, Berlin Heidelberg, pp 37–47

J.S. McCaskill (2001) Optically programming DNA computing in microflow reactors. *Biosystems* **59**, 125–138

A.P. Mills Jr. (2002) Gene expression profiling diagnosis through DNA molecular computation. *Trends Biotechnol.* **20**, 137–140

J. Minshull & W.P.C. Stemmer (1999) Protein evolution by molecular breeding. *Curr. Opin. Chem. Biol.* **3**, 284–290

J. Monod (1971) *Chance and necessity – on the natural philosophy of modern biology*. Penguin Books, London

G.E. Moore (1965) Cramming more components onto integrated circuits. *Electronics* **38**, 114–117

N. Morimoto, M. Arita & A. Suyama (1999) Solid phase DNA solution to the Hamiltonian Path Problem. In: H. Rubin & D.H. Wood (eds.) *DNA based computers III*. American Mathematical Society, Providence, RI, pp 193–206

T. Nakajima, Y. Sakai & A. Suyama (2002) Solving a 10-variable 43-clause instance of 3-SAT problems on DNA computer automatically executing a basic DNA instruction set. In: M. Hagiya & A. Ohuchi (eds.) *Preliminary proceedings of the 8th international meeting on DNA based computers, June 10 - 13 2002, Hokkaido University*. Sapporo, Japan, pp 332 (poster abstract)

A.J. Nataraj, I. Olivos-Glander, N. Kusukawa & W.E. Highsmith (1999) Single-strand conformation polymorphism and heteroduplex analysis for gel-based mutation detection. *Electrophoresis* **20**, 1177–1185

I. Nazarenko, R. Pires, B. Lowe, M. Obaidy & A. Rashtchian (2002) Effect of primary and secondary structure of oligodeoxyribonucleotides on the fluorescent properties of conjugated dyes. *Nucleic Acids Res.* **30**, 2089–2195

C.M. Niemeyer & M. Adler (2002) Nanomechanical devices based on DNA. *Angew. Chem. Int. Ed.* **41**, 3779–3783

D. Normile (2002) Molecular computing – DNA-based computer takes aim at genes. *Science* **295**, 951–951

M. Ogihara & A. Ray (1997) DNA-based parallel computing by 'counting'. In: H. Rubin & D.H. Wood (eds.) *DNA based computers III: DIMACS workschop, June 23-25, 1997*. American Mathematical Society, Providence, RI, pp 265–274

C.A. Oleykowski, C.R. Bronson Mullins, A.K. Godwin & A.T. Yeung (1998) Mutation detection using a novel plant endonuclease. *Nucleic Acids Res.* **26**, 4597–4602

J.S. Oliver (1997) Matrix multiplication with DNA. *J. Mol. Evol.* **45**, 161–167

Q. Ouyang, P.D. Kaplan, S.M. Liu & A. Libchaber (1997) DNA solution of the maximal clique problem. *Science* **278**, 446–449

H. Ørum, P.E. Nielsen, M. Egholm, R.H. Berg, O. Buchardt & C. Stanley (1993) Single base pair mutation analysis by PNA directed PCR clamping. *Nucleic Acids Res.* **21**, 5332–5336

P.A. Packan (1999) Pushing the limits. *Science* **285**, 2079–2081

G. Păun, G. Rozenberg & A. Salomaa (1998) *DNA computing: new computing paradigms*. Springer-Verlag, Berlin Heidelberg

G. Păun (2001) From cells to computers: computing with membranes (P systems). *Biosystems* **59**, 139–158

G. Păun & G. Rozenberg (2002) A guide to membrane computing. *Theor. Comput. Sci.* **287**, 73–100

N. Peyret, P.A. Seneviratne, H.T. Allawi & J. SantaLucia (1999) Nearest-neighbor thermodynamics and NMR of DNA sequences with internal A·A, C·C, G·G, and T·T mismatches. *Biochemistry* **38**, 3468–3477

M.J. Pérez-Jiménez & F. Sancho-Caparrini (2002) Solving knapsack problems in a sticker based model. In: N. Jonoska & N.C. Seeman (eds.) *DNA computing, 7th international meeting on DNA based computers*. Springer-Verlag, Berlin Heidelberg, pp 161–171

N.A. Pierce & E. Winfree (2002) Protein design is NP-hard. *Protein Eng.* **15**, 779–782

M.C. Pirrung, R.V. Connors, A.L. Odenbaugh, M.P. Montague-Smith, N.G. Walcott & J.J. Tollett (2000) The arrayed primer extension method for DNA microchip analysis. Molecular computation of Satisfaction problems. *J. Am. Chem. Soc.* **122**, 1873–1882

D. Pisinger (2004) Where are the hard knapsack problems? *Comput. Oper. Res., in press* (http://www.sciencedirect.com)

D.M. Prescott & G. Rozenberg (2002) How ciliates manipulate their own DNA – a splendid example of natural computing. *Nat. Comput.* **1**, 165–183

A. Radzicka & R. Wolfenden (1995) A proficient enzyme. *Science* **267**, 90–93

J.H. Reif (1998) Paradigms for biomolecular computation. In: C.S. Calude, J. Casti & M.J. Dinneen (eds.) *Unconventional models of computation*. Springer-Verlag, Berlin Heidelberg, pp 72–93

J.H. Reif & T.H. LaBean (2001) Computationally inspired biotechnologies: improved DNA sythesis and associative search using error-correcting codes and vector-quantization. In: A. Condon & G. Rozenberg (eds.) *DNA computing, 6th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 145–172

J.H. Reif, T.H. LaBean, M. Pirrung, V.S. Rana, B. Guo, C. Kingsford & G.S. Wickham (2002) Experimental construction of very large scale DNA databases with associative search capability. In: N. Jonoska & N.C. Seeman (eds.) *DNA computing, 7th international meeting on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 231–247

P.W.K. Rothemund (1995) A DNA and restriction enzyme implementation of Turing machines. In: E.B. Baum & R.J. Lipton (eds.) *DNA based computers, DIMACS 27*. American Mathematical Society, Providence, RI, pp 75–120

P.W.K. Rothemund, N. Papadakis & E. Winfree (2004) Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol.* **2**, e424

S. Roweis, E. Winfree, R. Burgoyne, N.V. Chelyapov, M.F. Goodman, L.M. Adleman & P.W.K. Rothemund (1998) A sticker-based model for DNA computation. *J. Comput. Biol.* **5**, 615–629

G. Rozenberg & H. Spaink (2002) Preface. *Nat. Comput.* **1**, 1–2

G. Rozenberg & H. Spaink (2003) DNA computing by blocking. *Theor. Comput. Sci.* **292**, 653–665

Y. Sakakibara & A. Suyama (2000) Intelligent DNA chips: logical operation of gene expression profiles on DNA computers. In: A.K. Dunker, A. Konagaya, S. Miyano & T. Takagi (eds.) *Genome Informatics* 2000. Universal Academy Press, Tokyo, pp 33–42

Y. Sakakibara & T. Hohsaka (2003) *In vitro* translation-based computations. In: J. Chen & J. Reif (eds.) *Preliminary proceedings 9th international meeting on DNA-based computers, 1-4 June 2003, Madison, Wisconsin*. University of Wisconsin, Madison, Wisconsin, USA, pp 175–179

K. Sakamoto, D. Kiga, K. Komiya, H. Gouzu, S. Yokoyama, S. Ikeda, H. Sugiyama & M. Hagiya (1999) State transitions by molecules. *Biosystems* **52**, 81–91

K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori & M. Hagiya (2000) Molecular computation by DNA hairpin formation. *Science* **288**, 1223–1226

J. Sambrook & D.W. Russell (2001) *Molecular cloning: a laboratory manual*. Cold Spring Harbor, New York

J. SantaLucia Jr., H.T. Allawi & P.A. Seneviratne (1996) Improved nearest-neighbor parameters for predicting DNA duplex stability. *Biochemistry* **35**, 3555–3562

J. SantaLucia Jr. (1998) A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci. USA* **95**, 1460–1465

H. Schagger & G. von Jagow (1987) Tricine sodium dodecyl-sulfate polyacrylamide-gel electrophoresis for the separation of proteins in the range from 1-kDa to 100-kDa. *Anal. Biochem.* **166**, 368–379

T.D. Schneider (1991) Theory of molecular machines. II. Energy dissipation from molecular machines. *J. Theor. Biol.* **148**, 125–137

T.D. Schneider (1994) Sequence logos, machine/channel capacity, Maxwell's demon, and molecular computers: a review of the theory of molecular machines. *Nanotechnology* **5**, 1–18

E. Schrödinger (1944) *What is life? – the physical aspect of the living cell*. Reprinted with *Mind and matter* and *autobiographical sketches* (1992), Cambridge University Press, Cambridge

P. Schwille, F.J. Meyer-Almes & R. Rigler (1997) Dual-color fluorescence cross-correlation spectroscopy for multicomponent diffusional analysis in solution. *Biophys. J.* **72**, 1878–1886

N.C. Seeman (1999) DNA engineering and its application to nanotechnology. *Trends Biotechnol.* **17**, 437–443

N.C. Seeman (2002) It started with Watson and Crick, but it sure didn't end there: pitfalls and possibilities beyond the classic double helix. *Nat. Comput.* **1**, 53–84

N.C. Seeman (2003) DNA in a material world. *Nature* **421**, 427–431

C.A.M. Seidel, A. Schulz & M.H.M. Sauer (1996) Nucleobase-specific quenching of fluorescent dyes. 1. Nucleobase one-electron redox potentials and their correlation with static and dynamic quenching efficiencies. *J. Phys. Chem.* **100**, 5541–5553

W.M. Shih, J.D. Quispe & G.F. Joyce (2004) A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron. *Nature* **427**, 618–621

M.L. Simpson, G.S. Sayler, J.T. Fleming & B. Applegate (2001) Whole-cell biocomputing. *Trends Biotechnol.* **19**, 317–323

L.M. Smith, R.M. Corn, A.E. Condon, M.G. Lagally, A.G. Frutos, Q. Liu & A.J. Thiel (1998) A surface-based approach to DNA computing. *J. Comp. Biol.* **5**, 255–267

R.M. Smith & D.E. Hansen (1998) The pH-rate profile for the hydrolysis of a peptide bond. *J. Am. Chem. Soc.* **120**, 8910–8913

C. Staehelin, C. Charon, T. Boller, M. Crespi & A. Kondorosi (2001) *Medicago truncatula* plants overexpressing the early nodulin gene *enod40* exhibit accelerated mycorrhizal colonization and enhanced formation of arbuscules. *Proc. Natl. Acad. Sci. USA* **98**, 15366–15371

W.P.C. Stemmer (1994) Rapid evolution of a protein *in vitro* by DNA shuffling. *Nature* **370**, 389–391

W.P.C. Stemmer (1995) The evolution of molecular computation. *Science* **270**, 1510

M.N. Stojanovic & D. Stefanovic (2003) A deoxyribozyme-based molecular automaton. *Nat. Biotechnol.* **21**, 1069–1074

E. Stoschek, M. Sturm & T. Hinze (2001) DNA-computing – ein funktionales Modell im laborpraktischen Experiment. *Informatik Forsch. Entw.* **16**, 35–52

X. Su & L.M. Smith (2004) Demonstration of a universal surface DNA computer. *Nucleic Acids Res.* **32**, 3115–3123

Y. Takenaka & A. Hashimoto (2003) Shortening the computational time of the fluorescent DNA computing. In: M. Hagiya & A. Ohuchi (eds.) *DNA computing, 8th international workshop on DNA-based computers*. Springer-Verlag, Berlin Heidelberg, pp 85–94

G.R. Taylor (1999) Enzymatic and chemical cleavage methods. *Electrophoresis* **20**, 1125–1130

J.M. Tour (2000) Molecular electronics. Synthesis and testing of components. *Acc. Chem. Res.* **33**, 791–804

A. Tsuji, H. Koshimoto, Y. Sato, M. Hirano, Y. Sei-lida, S. Kondo & K. Ishibashi (2000) Direct observation of specific messenger RNA in a single living cell under a fluorescence microscope. *Biophys. J.* **78**, 3260–3274

R.S. Tuma, M.P. Beaudet, X. Jin, L.J. Jones, C.Y. Cheung, S. Yue & V.L. Singer (1999) Characterization of SYBR gold nucleic acid gel stain: a dye optimized for use with 300-nm ultraviolet transilluminators. *Anal. Biochem.* **268**, 278–288

S. Tyagi, D.P. Bratu & F.R. Kramer (1998) Multicolor molecular beacons for allele discrimination. *Nat. Biotechnol.* **16**, 49–52

D.A. Upchurch, R. Shankarappa & J.I. Mullins (2000) Position and degree of mismatches and the mobility of DNA heteroduplexes. *Nucleic. Acids Res.* **28**, e69

J. Vieira & J. Messing (1991) New pUC-derived cloning vectors with different selectable markers and DNA-replication origins. *Gene* **100**, 189–194

L. Wang, Q. Liu, R.M. Corn, A. Condon & L.M. Smith (2000) Multiple word DNA computing on surfaces. *J. Am. Chem. Soc.* **122**, 7435–7440

L. Wang, J.G. Hall, M. Lu, Q. Liu & L.M. Smith (2001) A DNA computing readout operation based on structure-specific cleavage. *Nat. Biotechnol.* **19**, 1053–1059

R. Weiss, G.E. Homsy & T.F. Knight Jr. (2002) Toward *in vivo* digital circuits. In: L.F. Landweber & E. Winfree (eds.) *Evolution as computation*. Springer-Verlag, Berlin Heidelberg, pp 275–295

R. Weiss, S. Basu, S. Hooshangi, A. Kalmbach, D. Karig, R. Mehreja & I. Netravali (2003) Genetic circuit building blocks for cellular computation, communications, and signal processing. *Nat. Comput.* **2**, 47–84

S. Weiss (1999) Fluorescence spectroscopy of single biomolecules. *Science* **283**, 1676–1683

J.G. Wetmur (1991) DNA probes: applications of the principles of nucleic acid hybridization. *Crit. Rev. Biochem. Mol.* **26**, 227–259

K.A. Williams, P.T.M. Veenhuizen, B.G. de la Torre, R. Eritja & C. Dekker (2002) Carbon nanotubes with DNA recognition. *Nature* **420**, 761

M.C. Williams, J.R. Wenner, I. Rouzina & V.A. Bloomfield (2001) Effect of pH on the overstretching transition of double-stranded DNA: evidence of force-induced DNA melting. *Biophys. J.* **80**, 874–881

D.S. Wilson & J.W. Szostak (1999) *In vitro* selection of functional nucleic acids. *Annu. Rev. Biochem.* **68**, 611–647

E. Winfree, F. Liu, L.A. Wenzler & N.C. Seeman (1998) Design and self-assembly of two-dimensional DNA crystals. *Nature* **394**, 539–544

D. Wood, J. Chen, E. Antipov, B. Lemieux & W. Cedeno (1999) A DNA Implementation of the Max 1s Problem. In: W. Banzhaf, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela & R.E. Smith (eds.) *Proceedings of the genetic and evolutionary computation conference 1999*. Morgan Kaufman, San Francisco, pp 1835–1841

H. Yan, X. Zhang, Z. Shen & N.C. Seeman (2002) A robust DNA mechanical device controlled by hybridization topology. *Nature* **415**, 62–65

H. Yan, T.H. LaBean, L. Feng & J.H. Reif (2003) Directed nucleation assembly of DNA tile complexes for barcode-patterned lattices. *Proc. Natl. Acad. Sci. USA* **100**, 8103–8108

B. Yang, X. Wen, N.S. Kodali, C.A. Oleykowski, C.G. Miller, J. Kulinski, D. Besack, J.A. Yeung, D. Kowalski & A.T. Yeung (2000) Purification, cloning, and characterization of the CEL I nuclease. *Biochemistry* **39**, 3533–3541

Y. Yokobayashi, R. Weiss & F.H. Arnold (2002) Directed evolution of a genetic circuit. *Proc. Natl. Acad. Sci. USA* **99**, 16587–16591

T. Yokomori (2002) Molecular computing paradigm – toward freedom from Turing's charm. *Nat. Comput.* **1**, 333–390

H. Yoshida & A. Suyama (2000) Solution to 3-SAT by breadth first search. In: E. Winfree & D.K. Gifford (eds.) *DNA based computers V*. American Mathematical Society, Providence, RI, pp 9–22

I.T. Young, R. Moerman, L.R. van den Doel, V. Iordanoc, A. Kroon, H.R.C. Dietrich, G.W.K. Dedem, A. Bossche, B.L. Gray, L. Sarro, P.W. Verbeek & L.J. van Vliet (2003) Monitoring enzymatic reactions in nanoliter wells. *J. Microsc. -Oxf.* **212**, 254–267

B. Yurke, A.P. Mills Jr. & S.L. Cheng (1999) DNA implementation of addition in which the input strands are separate from the operator strands. *Biosystems* **52**, 165–174

B. Yurke, A.J. Turberfield, A.P. Mills Jr., F.C. Simmel & J.L. Neumann (2000) A DNA-fuelled molecular machine made of DNA. *Nature* **406**, 605–608

Curriculum vitae

De schrijver van dit proefschrift werd op 20 juli 1975 geboren in Den Haag, en in die stad behaalde hij in 1993 zijn VWO diploma aan het Christelijk Gymnasium Sorghvliet. De rest van de jaren '90 besteedde hij aan een studie biologie aan de Universiteit Leiden, met als wetenschappelijke hoogtepunten een onderzoek onder begeleiding van dr. Anne-Marie Zeeman en dr. ir. Yde Steensma aan het pyruvaatmetabolisme van gist, en een onderzoek onder begeleiding van prof. dr. Diedel Kornet in de grondslagen van de biologie.

In januari 2000 begon hij als promovendus aan het in dit proefschrift beschreven onderzoek – in dienst van het Leiden Institute of Advanced Computer Science (LIACS), maar met een zitkamer en postvak bij het Instituut Biologie van de Universiteit Leiden (IBL). Het sterk interdisciplinaire karakter van het onderzoek naar evolutionaire DNA computers komt ook tot uiting in de professionele diversiteit van de betrokken begeleiders: prof. dr. Herman Spaink (IBL, moleculaire celbiologie), prof. dr. Grzegorz Rozenberg (LIACS, theoretische informatica), prof. dr. Joost Kok (LIACS, fundamentele informatica) en prof. dr. Thomas Bäck (LIACS, natural computing).