DNASequenceGenerator: A Program for the Construction of DNA Sequences

Udo Feldkamp[1], Sam Saghafi[2], Wolfgang Banzhaf[1], and Hilmar Rauhe[1]

[1] University of Dortmund, Chair of System Analysis, Germany,
{feldkamp, banzhaf, rauhe}@LS11.cs.uni-dortmund.de,
http://ls11-www.informatik.uni-dortmund.de/molcomp/
[2] University of Cologne, Institute of Genetics, Germany

Abstract. In DNA Computing and DNA nanotechnology the design of proper DNA sequences turned out to be an elementary problem [1, 2, 3, 4, 5, 6, 7, 8, 9]. We here present a software program for the construction of sets ("pools") of DNA sequences. The program can create DNA sequences to meet logical and physical parameters such as uniqueness, melting temperature and GC ratio as required by the user. It can create sequences *de novo*, complete sequences with gaps and allows import and recycling of sequences that are still in use. The program always creates sequences that are — in terms of uniqueness, GC ratio and melting temperature — "compatible" to those already in the pool, no matter whether those were added manually or created or completed by the program itself. The software comes with a GUI and a Sequence Wizard. *In vitro* tests of the program's output were done by generating a set of oligomers designed for self-assembly. The software is available for download under http://LS11-www.cs.uni-dortmund.de/molcomp/Downloads/downloads.html.

1 Introduction

The most important requirement for DNA sequences useful for computation is the avoidance of non-specific hybridizations. These can occur between the sequences used in a self-assembly step, in a polymerase chain reaction, in an extraction operation etc. Thus the main purpose for designing DNA sequences is finding a set of sequences as dissimilar as possible, where the dissimilarity usually includes comparison to complementary sequences. Another important aspect is the control of the thermodynamic properties of the sequences, allowing the design of a protocol (the actual application) minimizing the probability of hybridization errors and possibly regarding other constraints given by the application.

There are several approaches to DNA sequence design. Seeman et al. designed sequences using overlapping subsequences to enforce uniqueness [1, 2]. The approach is based on the "repairing" of sequences. Deaton et al. used genetic algorithms to generate a set of unique DNA sequences using the Hamming distance for measuring the uniqueness [3, 4]. Marathe, Condon and Corn chose a dynamic programming approach for DNA sequence design, also using the Hamming distance [5]. They also described a dynamic programming based algorithm for the selection of sequences with a given free energy. Frutos et al. developed a so-called template-map strategy to get a grand number of dissimilar sequences while having to design only a significantly smaller number of templates and maps [6]. They also use a Hamming-like dissimilarity with no shifts of the regarded sequences. Hartemink, Gifford and Khodor designed sequences for the programmed mutagenesis, which demands similar sequences with only a few mismatches [7]. The selection of appropriate sequences is done by exhaustive search, which is feasible for short oligomers. Faulhammer et al. described a designing algorithm for RNA sequences to be used in solving a chess problem [8]. They also use the Hamming distance as measurement for uniqueness and do not construct the sequences but repair them as long as necessary, potentially non-terminating. Baum suggested a method to design unique sequences by avoiding multiple usage of subsequences by restricting the choice of nucleotides at the ends of the sequences [9].

2 Theoretical Background

The program described here uses a concept of uniqueness that, within a pool of sequences, allows any subsequence of a certain (definable) length to occur at most once in that pool. This concept of uniqueness is related to the one described by Seeman et al. [1, 2] but a different approach was chosen. In particular the software described here uses a fully automatic, graph-based approach [10].

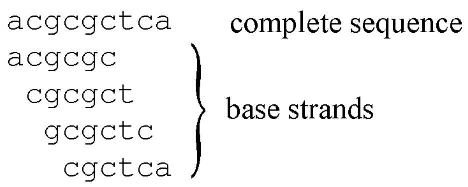


Fig. 1. A sequence of length $n_\mathrm{s} = 9$ consisting of $(n_\mathrm{s} - n_\mathrm{b} + 1) = 4$ overlapping base strands of length $n_\mathrm{b} = 6$.

According to this concept a pool of sequences is said to be $n_\mathrm{b} - \mathit{unique}$ if any subsequence in the pool of length n_b is unique, i.e. all sequences of the pool have common substrings of maximum length $n_\mathrm{b} - 1$. Uniqueness on the other hand is defined as $1 - (n_\mathrm{b} - 1)/n_\mathrm{s}$, a ratio to measure how much of a set of sequences of length n_s is unique. For example, 20-mers that are 10-unique have common subsequences of at most 9 subsequent nucleotides and a uniqueness of 55%.

The generation algorithm uses a directed graph, where the nodes are base strands (the unique strands of minimal length n_b) and the successors of a node are those four strands that can appear as an (overlapping) successor in a longer sequence (see Fig. 1, 2). Thus, a set of n_b-unique sequences of length n_s corresponds to a set of paths of length $(n_\mathrm{s} - n_\mathrm{b} + 1)$ through this graph having no node in common.

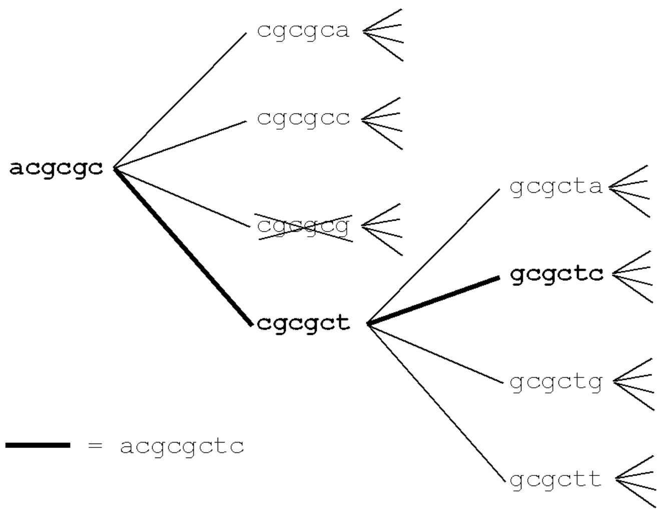


Fig. 2. Graph of base strands. A path of m nodes represents a sequence of length $n_\mathrm{b} + m - 1$. The node `cgcgcg` is self-complementary and therefore not used.

The details of this algorithm have been described in more detail earlier [10]. Note that this concept of uniqueness restricts the number of usable sequences strictly. The number of base strands of length n_b is

$$N_{bs}(n_\mathrm{b}) = 4^{n_\mathrm{b}} \quad . \tag{1}$$

Since complements of already used base strands are not used themselves, self-complementary base strands are not used at all. The number of base strands that can be used in the generation process is

$$N_{useful}(n_\mathrm{b}) = \frac{N_\mathrm{bs}(n_\mathrm{b}) - 4^{n_\mathrm{b}/2}}{2} \tag{2}$$

if n_b is even and

$$N_{useful}(n_\mathrm{b}) = \frac{N_\mathrm{bs}(n_\mathrm{b})}{2} \tag{3}$$

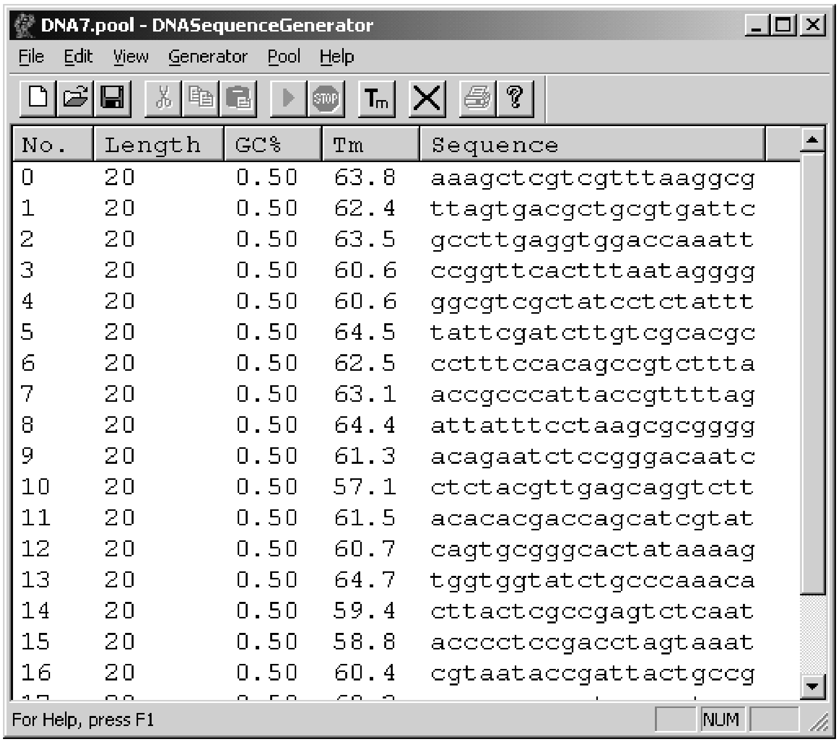
if n_b is odd, because there are no self-complementary base strands of odd length. A sequence of length n_s consists of $n_\mathrm{s} - n_\mathrm{b} + 1$ base strands. Thus, the maximum number of sequences built with the described algorithm is

$$N_{seqs}(n_\mathrm{s}, n_\mathrm{b}) = \left\lfloor \frac{N_\mathrm{useful}(n_\mathrm{b})}{n_\mathrm{s} - n_\mathrm{b} + 1} \right\rfloor . \tag{4}$$

This estimation is probably a bit too high because it does not include the constraint of the base strands having to overlap. Further requirements such as GC-ratio, melting temperature, the exclusion of long guanine subsequences or of start codons decrease the yield.

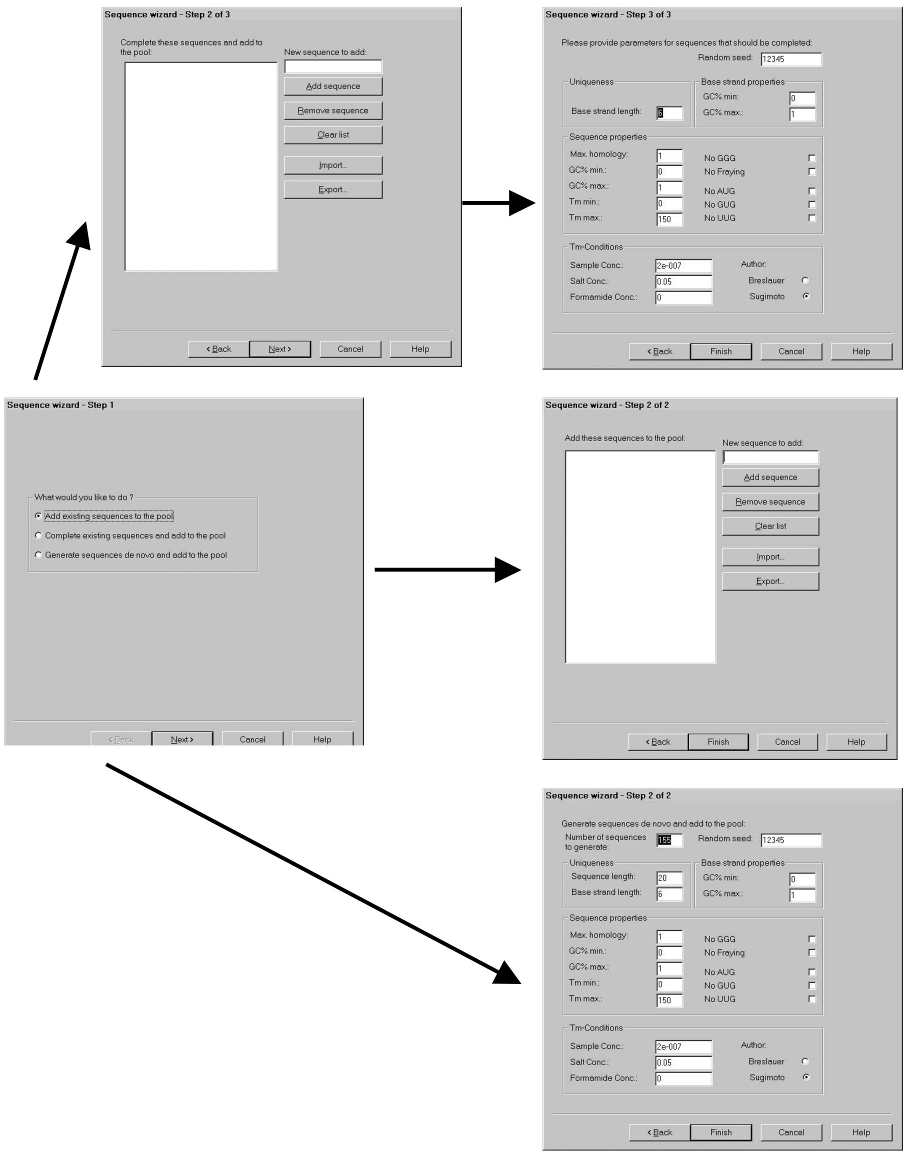
3 DNA Sequence Generator

The program DNASequenceGenerator is based on the metaphor of a pool of sequences that can be iteratively filled with sequences which meet the logical and physical requirements. The main window represents a pool of sequences (Fig. 3). The user can add, import, export and print sequences to and from it. E.g., one might import sequences already available in the lab and add new sequences that are compatible to those in terms of uniqueness, melting temperature and GC-ratio.


DNA7.pool - DNASequenceGenerator
File Edit View Generator Pool Help
For Help, press F1
NUM

No.	Length	GC%	Tm	Sequence
0	20	0.50	63.8	aaagctcgtcgtttaaggcg
1	20	0.50	62.4	ttagtgacgctgcgtgattc
2	20	0.50	63.5	gccttgaggtggaccaaatt
3	20	0.50	60.6	ccggttcactttaatagggg
4	20	0.50	60.6	ggcgtcgctatcctctattt
5	20	0.50	64.5	tattcgatcttgtcgcacgc
6	20	0.50	62.5	cctttccacagccgtcttta
7	20	0.50	63.1	accgcccattaccgttttag
8	20	0.50	64.4	attatttcctaagcgcgggg
9	20	0.50	61.3	acagaatctccgggacaatc
10	20	0.50	57.1	ctctacgttgagcaggtctt
11	20	0.50	61.5	acacacgaccagcatcgtat
12	20	0.50	60.7	cagtgcgggcactataaaag
13	20	0.50	64.7	tggtggtatctgcccaaaca
14	20	0.50	59.4	cttactcgccgagtctcaat
15	20	0.50	58.8	acccctccgacctagtaaat
16	20	0.50	60.4	cgtaataccgattactgccg

Fig. 3. Screenshot of the main window, containing a pool of sequences.



Sequence wizard - Step 1

What would you like to do ?
- [✓ Add existing sequences to the pool]
- Complete existing sequences and add to the pool
- Generate sequences de novo and add to the pool

Sequence wizard - Step 2 of 3

Complete these sequences and add to the pool:

New sequence to add:

Sequence wizard - Step 3 of 3

Please provide parameters for sequences that should be completed:

Random seed: 12345

Uniqueness — Base strand length: 6

Base strand properties — GC% min: 0; GC% max.: 1

Sequence properties — Max. homology: 1; GC% min.: 0; GC% max.: 1; Tm min.: 0; Tm max.: 150; No GGG ☐; No Fraying ☐; No AUG ☐; No GUG ☐; No UUG ☐

Tm-Conditions — Sample Conc.: 2e-007; Salt Conc.: 0.05; Formamide Conc.: 0; Author: Breslauer [✓ Sugimoto]

Sequence wizard - Step 2 of 2

Add these sequences to the pool:

New sequence to add:

Sequence wizard - Step 2 of 2

Generate sequences de novo and add to the pool:

Number of sequences to generate: 155

Random seed: 12345

Uniqueness — Sequence length: 20; Base strand length: 6

Base strand properties — GC% min: 0; GC% max.: 1

Sequence properties — Max. homology: 1; GC% min.: 0; GC% max.: 1; Tm min.: 0; Tm max.: 150; No GGG ☐; No Fraying ☐; No AUG ☐; No GUG ☐; No UUG ☐

Tm-Conditions — Sample Conc.: 2e-007; Salt Conc.: 0.05; Formamide Conc.: 0; Author: Breslauer [✓ Sugimoto]

Fig. 4. The different steps of the sequence wizard for the three possible ways to add sequences to the pool.

The process of constructing sequences is controlled by using a "Sequence Wizard" (Fig. 4). The sequence wizard enables the user to:

1. import or manually add existing or strictly required sequences
2. import or manually add sequence templates that are completed to full sequences if possible. Sequence templates use a simple notation: The preset

nucleotides (e.g. of a restriction site or other functional subsequence) are specified normally, while an `n` stands for the positions to fill. E.g., if given the sequence template `nnnnaacgttnnnn`, the generator replaces the leading and rear four `n`s with nucleotides.
3. generate sequences *de novo*.

A pool of sequences can be built iteratively invoking the sequence wizard repeatedly using different parameter sets.

The user can also use the DNASequenceGenerator as a T_m calculator for whole sequence pools. After importing the sequences to the pool the melting temperature is calculated automatically for each sequence. After changing the pool conditions, T_m will be re-calculated for all sequences in the pool. Pool conditions that can be parameterized by the user are sample concentration, monoionic salt concentration and formamide concentration. Also the method of estimating the melting temperature can be chosen (Wallace rule, GC-% formula, nearest-neighbor method) as well as different parameter sets for the nearest-neighbor method.

4 Results

4.1 In Silico

Experiments *in silico* were made to examine the possible yield of n_b-unique DNA sequences (Table 1). Ten runs with different random number generator seeds were made for each combination of n_b- and n_s-values, which ranged from 4 to 7 nt (for n_b) and 10 to 40 nt (for n_s), respectively. Most runs achieved a yield of 80–90 % of the theoretically estimated maximum number of sequences (see (4)).

4.2 In Vitro

In order to test the program's output, oligonucleotides for parallel overlap assembly have been generated and assembled *in vitro*. The single-stranded molecules overlap by 20 nucleotides (E/O sections, see Fig. 5), where the overlap assembly takes place, and a core sequence that stays single stranded in the assembly step and is filled later by the use of polymerase. Additionally, the sequences had to have restriction sites at specified locations.

The DNASequenceGenerator provided the needed sequences, which hybridised which high specificity to form the desired molecules.

Additionally, the melting temperature of a batch of 51 of the generated sequences with a length of 20 bp and a GC ratio of 50 % was analyzed with a Roche LightCycler^{TM} (Figs. 6, 7). Since the oligos were selected for their GC ratios rather than their melting temperatures, their melting temperatures ranged between 53 and 63 °C.

Table 1. Yield of n_b-unique sequences averaged over 10 runs, theoretic maximum yield and ratio. This is only an excerpt of all experimental results.

n_s	$n_b = 4$	$n_b = 5$	$n_b = 6$	$n_b = 7$
15	8.1 of 10 (81.0 %)	38.6 of 46 (83.9 %)	165.4 of 201 (82.3 %)	757.8 of 910 (83.3 %)
16	7.3 of 9 (81.1 %)	35.6 of 42 (84.8 %)	150.5 of 183 (82.2 %)	682.1 of 819 (83.3 %)
17	6.8 of 8 (85.0 %)	33.0 of 39 (84.6 %)	138.5 of 168 (82.4 %)	623.4 of 744 (83.8 %)
18	6.1 of 8 (76.3 %)	30.7 of 36 (85.3 %)	127.3 of 155 (82.1 %)	573.3 of 682 (84.1 %)
19	5.8 of 7 (82.9 %)	28.4 of 34 (83.5 %)	118.4 of 144 (82.2 %)	531.1 of 630 (84.3 %)
20	5.6 of 7 (80.0 %)	26.6 of 32 (83.1 %)	111.4 of 134 (83.1 %)	494.8 of 585 (84.6 %)
30	3.3 of 4 (82.5 %)	16.9 of 19 (88.9 %)	67.8 of 80 (84.8 %)	294.0 of 341 (86.2 %)
40	2.0 of 3 (66.7 %)	12.2 of 14 (87.1 %)	48.2 of 57 (84.6 %)	211.1 of 240 (88.0 %)

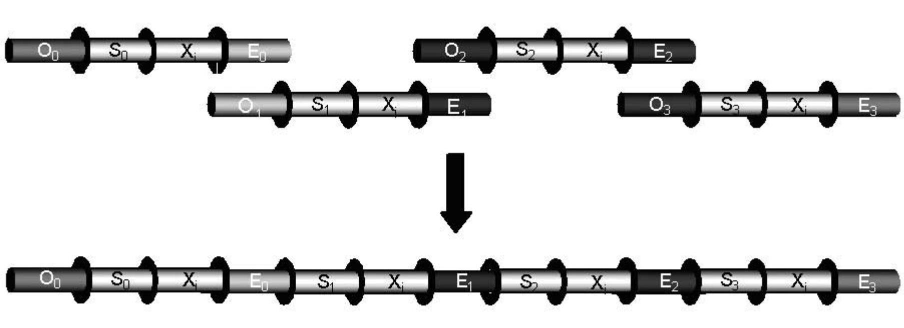


Fig. 5. Overlap assembly of DNA sequences. The O_i- and E_i- subsequences were designed such that E_i is complementary to O_{i+1}.

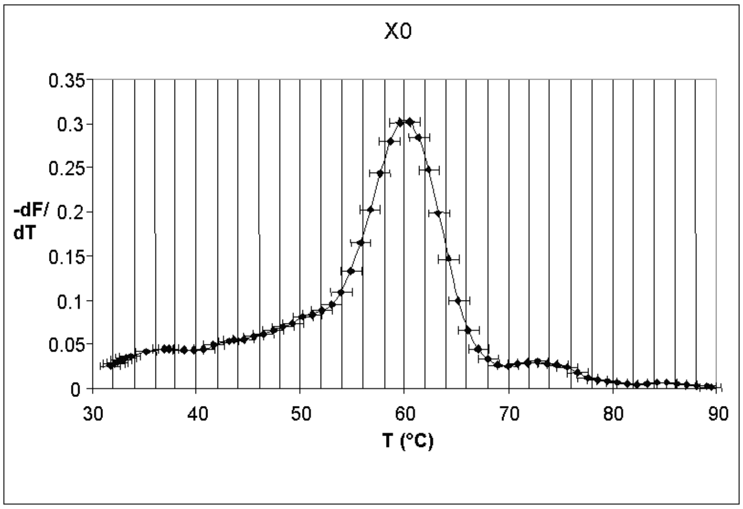


Fig. 6. Sample melting plot of one of the oligos. The y-axis shows the fluorescence absorption, the x-axis shows the current temperature in °C. In this case the oligo's T_m is 60.1 °C.

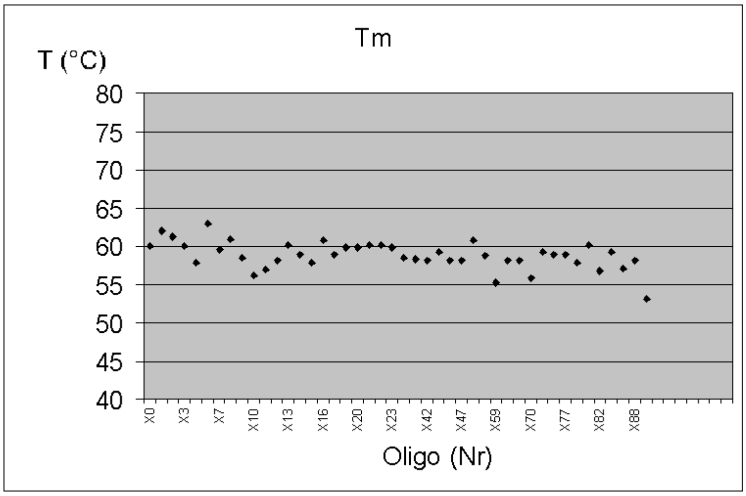


Fig. 7. Distribution of T_m's of 51 oligonucleotides. All oligonucleotides were 20 bp long and had 50 % GC ratio. The distribution of T_m's is in a range from 53 to 63 °C.

5 One Step Further: The DNASequenceCompiler

While the DNASequenceGenerator could, in principle, be used for sticky-end design, its sibling tool, the DNASequenceCompiler, is more specifically suited for this task [10]. It is designed to translate formal grammars directly into DNA molecules representing the rules of the grammar. As these rules determine how terminals are assembled to expressions, they also determine how the DNA molecules self-assemble to larger molecules. Thus, the DNASequenceCompiler provides an interface for the programmable self-assembly of molecules.

It mainly consists of three parts: a parser module for reading the "source code" which contains the symbol sets and the rules of the grammar as well as physical of chemical requirements for the sequences; the generator as a core module for the generation of the sequences; and a coordinating instance controlling the use of the generator while regarding the additional requirements in respect to uniqueness that arise for concatenated sequences.

Currently, a preliminary version of this tool is in use for the design of sequences for algorithmic self-assembly. A user-friendly version is under development. Further enhancements will tackle branched molecules, the consideration of secondary structures occurring on purpose, and a "programming language" on a higher level of abstraction than a formal grammar.

6 Conclusion

Here, a software tool for the design of DNA oligomers useful for DNA computation and DNA Nanotechnology was presented. The software uses a graph-based approach and generates pools of unique DNA sequences automatically according to the user's logical and physical requirements. First experimental results with sequences yielded with the software are encouraging and suggest further investigation. Additional experiments to measure melting temperatures, uniqueness of the sequences and specific hybridization behaviour are currently under investigation. A challenging task will be the development of a benchmark protocol.

The DNASequenceGenerator can be seen as a basic design tool for multiple purposes. The design of sophisticated DNA structures, such as cubes [11] or double- [12] or triple-crossover molecules [13] will require a more specialized program. The DNASequenceCompiler which is currently developed is intended for such a purpose and may help design molecules suitable for algorithmic self-assembly.

7 Acknowledgements

We would like to thank Jonathan C. Howard and the members of his group from the Institute for Genetics, University of Cologne, for their friendly support. The work was supported in parts by the Stifterverband der Deutschen Wissenschaft/Stiftung Winterling Markleuthen.

References

[1] Seeman, N. C., Kallenbach, N. R.: Design of immobile Nucleic Acid Junctions. Biophysical Journal **44** (1983) 201–209

[2] Seeman, N. C.: De Novo Design of Sequences for Nucleic Acid Structural Engineering. Journal of Biomolecular Structure & Dynamics **8**(3) (1990) 573–581

[3] Deaton, R., Murphy, R. C., Garzon, M., Franceschetti, D. T., Stevens Jr., S. E.: Good Encodings for DNA-based Solutions to Combinatorial Problems. Proceedings of the Second Annual Meeting on DNA Based Computers, held at Princeton University (1996) 159–171

[4] Deaton, R., Murphy, R. C., Rose, J. A., Garzon, M., Franceschetti, D. T., Stevens Jr., S. E.: Genetic Search for Reliable Encodings for DNA-based Computation. First Conference on Genetic Programming (1996)

[5] Marathe, A., Condon, A. E., Corn, R. M.: On Combinatorial DNA Word Design. Proceedings of the 5th International Meeting on DNA Based Computers (1999)

[6] Frutos, A. G., Liu, Q., Thiel, A. J., Sanner, A. M. W., Condon, A. E., Smith, L. M., Corn, R. M.: Demonstration of a Word Design Strategie for DNA Computing on Surfaces. Nucleic Acids Research **25**(23) (1997) 4748–4757

[7] Hartemink, A. J., Gifford, D. K., Khodor, J.: Automated Constraint-Based Nucleotide Sequence Selection for DNA Computation. Proceedings of the 4th DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, Philadelphia (1998) 227–235

[8] Faulhammer, D., Cukras, A. R., Lipton, T. J., Landweber, L. F.: Molecular Computation: RNA Solutions to Chess Problems Proceedings of the National Acadamy of Sciences USA **97**(4) (2000) 1385–1389

[9] Baum, E. B.: DNA Sequences Useful for Computation. Unpublished, available under http://www.neci.nj.nec.com/homapages/eric/seq.ps (1996)

[10] Feldkamp, U., Banzhaf, W., Rauhe, H.: A DNA Sequence Compiler. Proceedings of the 6th DIMACS Workshop on DNA Based Computers, held at the University of Leiden, The Netherlands (2000) 253. Manuscript available at http://LS11-www.cs.uni-dortmund.de/molcomp/Publications/publications.html

[11] Chen J., Seeman, N. C.: Synthesis from DNA of a Molecule with the Connectivity of a Cube. Nature **350** (1991) 631–633

[12] Winfree, E., Liu, F., Wenzler, L. A., Seeman, N. C.: Design and Self-assembly of Two-dimensional DNA Crystals. Nature **394** (1998) 539–544

[13] Mao, C., LaBean, T. H., Reif, J. H., Seeman, N. C.: Logical Computation Using Algorithmic Self-assembly of DNA Triple-crossover Molecules. Nature **407** (2000) 493–496