EGNAS software manual

Alfred Kick* kick@ksi-meinsberg.de
Martin Bönsch† martin.boensch@tu-dresden.de
Michael Mertig*† michael.mertig@tu-dresden.de

June 25, 2012

1 Introduction

The exhaustive DNA sequence design algorithm EGNAS (Exhaustive Generation of Nucleic Acid Sequences) allows to generate sets containing a maximum number of sequences with defined properties. It offers the possibility of controlling both interstrand and intrastrand properties. The guanine-cytosine content can be adjusted. Sequences can be forced to start and end with guanine or cytosine. This option reduces the risk of "fraying" in DNA strands. It is possible to limit cross hybridizations of a defined length, and to adjust the uniqueness of sequences. Self-complementarity and hairpin structures of certain length can be avoided. Sequences and subsequences can be forbidden optionally. Furthermore, sequences can be designed to have minimum interactions with predefined strands and neighboring sequences.
TAG sequences can be generated and combined with primers for single base extension reactions, which were described for multiplexed genotyping of single nucleotide polymorphisms. Thereby, possible foldback through intrastrand interaction of TAG-primer pairs can be limited. The design of sequences for specific attachment of molecular constructs to DNA origami is possible.
The algorithm is realized in a C++ program. EGNAS is freely available for noncommercial use at `http://www.chm.tu-dresden.de/pc6/EGNAS`. A detailed description of EGNAS is published [1].

2 Running the program

The sequence design algorithm EGNAS is realized in a program written in C++. It is currently a command line program which was compiled for Linux, Mac OS X and Microsoft Windows operating systems. EGNAS is started by running the executable file after unpacking the ZIP file for the proper operating system. Forbidden subsequence motifs can be given in a text file named "forbidden.txt". This file and the text files with the included and neighboring sequences have to be saved in the folder of the executable file. All of those sequences have to be separated by a newline and consist of the characters from the set $\{A; T; C; G\}$. Additionally, a configuration file is needed to choose the sequence design criteria and options. If no configuration file "config.txt" exists in the current folder, the program will create the "config.txt".

3 Sequence design criteria and options

The sequence design algorithm EGNAS offers the user different options. Consequently, the generated sequences meet certain criteria:

1. Sequence length L_s.
2. Length of basic sequences (criton length) L_c.
3. Exact GC content or its range.

*Germany, Kurt-Schwabe-Institut für Mess- und Sensortechnik e.V. Meinsberg, Kurt-Schwabe-Straße 4, 04720 Ziegra-Knobelsdorf, `http://www.ksi-meinsberg.de`
†Professur für Physikalische Chemie, Mess- und Sensortechnik, Technische Universität Dresden, 01062 Dresden, Germany, `http://www.chm.tu-dresden.de/pc6`

4. No terminal adenine or thymine in the strand./Demand on “GC ends”.
5. Forbidden sequences./Included sequences.
6. Length of forbidden self-complementary subsequences L_{sc}.
7. Forbidden stem length of hairpin structures L_{hp}.
8. Length of subsequences that are not allowed to be repeated within one and the same sequence (“sliding”) L_{sl}.
9. Forbidden length of subsequences that could interact with complementary neighboring sequences L_{ni}.

The EGNAS software provides the option to calculate the molar free enthalpy of DNA duplex formation. This calculation is based on the nearest-neighbor model [2] with parameters taken from SantaLucia et al. [3].

3.1 Hairpins and self-complementarity

Hairpin structures are also called stem-and-loop structures. They consist of two complementary arm sequences and the loop sequence. The arm sequences are able to form the double-stranded stem while being connected by the single-stranded loop sequence. Self-complementarity is treated as a special case of a hairpin when the loop size is zero. Therefore, a self-complementary sequence has always an odd number of bases. If hairpin structures with a L_{hp} bases long stem are forbidden, self-complementary subsequences that are equal to or longer than $2L_{hp}$ bases will consequentially be omitted.

3.2 Sliding

We use the term “sliding” for an intrastrand property of a sequence. Sliding denotes that a subsequence can be found several times at different positions of one and the same strand. If a complementary strand hybridizes with such a strand, different positions will be possible. Thus, sliding between the hybridizing strands would take place. For example, the following sequence pair allows sliding due to the repetition of 7 bases long subsequences ($L_{sl} = 8$).

```
5'-ACGACGACGA-3'    5'-ACGACGACGA-3'    5'-ACGACGACGA-3'
      |||||||           ||||||||||          |||||||
  3'-TGCTGCTGCT-5' 3'-TGCTGCTGCT-5' 3'-TGCTGCTGCT-5'
```

3.3 Interaction with the neighboring sequences

If primers are paired with TAGs, a special criterion for the TAG sequences will arise. In this case, primer foldback can become a problem.
Interactions with neighboring strands are already limited during the sequence generation. Therefore, all possible L_{ni} bases long complementary subsequences of a neighboring strand are forbidden for the generation of the corresponding TAG. This is especially intended for designing strands where a molecular spacer is located between the neighboring sequence and the TAG. For example, such a spacer could be a hexaethylene glycol moiety. Actually, overlapping subsequences, which would evolve through directly attaching one neighboring sequence to either the 3’ or 5’ end of a TAG, are not considered. Nevertheless, even if TAGs are attached to neighboring sequences without an intended spacer, hairpin formation will still be diminished significantly.

3.4 The configuration file “config.txt”

The configuration file “config.txt” contains the settings for the sequence design. An example is shown below.

```
#Created with version: 1158
#Sequence length L_s.
L_s[bases]=24
#Length of basic sequences (criton length) 1 <= L_c <= 14.
L_c[bases]=5
#Forbidden stem length of hairpin structures L_hp (blank for any size).
L_hp[base pairs]=3
#Length of forbidden self-complementary subsequences L_sc (blank for any size).
L_sc[bases]=4
#Minimum number of G/C-bases per strand GC_min, minimum GC content (blank for
 any number).
GC_min[bases]=11
#Maximum number of G/C-bases per strand GC_max, maximum GC content (blank for
 any number).
GC_max[bases]=13
#Length of subsequences that are not allowed to be repeated within one and the
 same sequence (sliding) L_sl (blank for any size).
L_sl[bases]=5
#No terminal adenine or thymine in the strand?/Demand on GC ends? [y/n]
GC_ends=n
#Maximum number of sequences to be generated per set.
SeqMax=1250
#Maximum number of sets to be generated.
SetMax=2
#Maximum number of attempts to restart the generation of one sequence.
MaxAttempt=50
#Termination factor to limit the number of combination attempts with basic
 sequences for the generation of one strand.
TerminationFactor=1
#Name of the file with sequences to be included (blank, if no file exist).
Included=
#Name of the file with neighboring sequences (blank, if no file exist).
Neighboring=
#Forbidden stem length of hairpin structures with neighboring sequences L_ni
 (neighbor interaction).
L_ni[base pairs]=
#Calculate deltaG? [y/n]
deltaG=n
```

4 The output of the results

The output of the results is saved in a separate folder. Its name is the current date and system time. A log file contains the settings and further information about the sequence generations. Every generated set of sequences is saved in a separate text file.

5 Availability and requirements

Project name: EGNAS

Project home page: `http://www.chm.tu-dresden.de/pc6/EGNAS`

Operating systems: Linux, Mac OS X, and Microsoft Windows

Programming language: C++

Other requirements: None

License: Free for noncommercial use

Any restrictions to use by nonacademics: License needed

References

[1] A. Kick, M. Bönsch, M. Mertig, EGNAS: An exhaustive dna sequence design algorithm, BMC Bioinformatics (2012) 13:138doi:10.1186/1471-2105-13-138.

[2] K. J. Breslauer, R. Frank, H. Blöcker, L. A. Marky, Predicting DNA duplex stability from the base sequence, Proceedings of the National Academy of Sciences 83 (11) (1986) 3746–3750.

[3] J. SantaLucia, Jr., H. T. Allawi, P. A. Seneviratne, Improved nearest-neighbor parameters for predicting DNA duplex stability, Biochemistry 35 (11) (1996) 3555–3562.