A global heuristically search algorithm for DNA encoding*

Zhang Kai**, Pan Linqiang and Xu Jin

(The Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

Accepted on November 30, 2006

Abstract A new efficient algorithm is developed to design DNA words with equal length for DNA computing. The algorithm uses a global heuristic optimizing search approach and converts constraints to a carry number to accelerate the convergence, which can generate a DNA words set satisfying some thermodynamic and combinatorial constraints. Based on the algorithm, a software for DNA words design is developed.

Keywords: DNA computing, encoding, heuristic optimizing search.

Since Adleman[1] and Lipton[2] demonstrated the possibility of solving NP-complete problems[3] by using DNA, DNA computing has become a new vista of computation that bridges computer science and biochemistry. Since each piece of information is encoded in biological sequences, their design is crucial for successful DNA computation. DNA computing that has great potential to solve complex problems has invoked interests in DNA words generator algorithm.

For a set of DNA words to be effective for the DNA computing, they must fulfill a number of combinatorial and thermodynamic constraints. Of particular importance is the need for specific hybridization between a given word and its unique Watson-Crick complement. For this requirement, Frutos et al. proposed the template method for DNA word design[4,5]. Feldkamp demonstrated a DNA sequence complier algorithm for designing DNA sequences[6]. Reaton presented a genetic algorithm for generating DNA strands[7—9]. Obviously, there is a tradeoff between the tightness of the constraints and the number of sequences which we can design. These methods do not guarantee the optimality in terms of the number of sequences which can be designed.

In this paper, a new algorithm is described, which uses a global heuristically search method. The combinatorial and thermodynamic design criteria are incorporated into our algorithm. Due to all the constraints in our algorithm are parallel, additional constraints are supported by the algorithm and can be integrated into our model in a straightforward way. Because the algorithm is based on global search, the DNA sequence set is one of greatest sets which satisfy the constraints. Our algorithm has been implemented on Pascal language complier of Borland Delphi 7.

1 Design constraints

Let $X = x_1 x_2 \cdots x_n$ be a word, where x_i belongs to an alphabet. In this paper, we deal with two alphabets, the quaternary alphabet $\{0, 1, 2, 3\}$ and the DNA alphabet $\{A, C, G, T\}$. The elements of an alphabet are called characters. The goal of the algorithm is to design DNA words, but in intermediate step our algorithm generates some quaternary words.

Our algorithm can design a set S of equally long DNA strands, being either words or complements. In this section, we list the combinatorial and thermodynamic constraints that are supported by our algorithm. The user of the algorithm may choose the constraints by their need during the design of DNA words. In this paper, we introduce four kinds of common constraints, and demonstrate how to integrate these constraints into our algorithm. In addition, other constraints can be integrated into the algorithm in a simple and straightforward way.

1.1 GC content constraint

The GC content is the number of bases in any

* Supported by National Natural Science Foundation of China (Grant Nos. 60533010, 60373089, and 60674106), the Ph. D. Programs Foundation of Ministry of Education of China (No. 20060487014) and Program for New Century Excellent Talents in University (No. NCET-05-0612)

** To whom correspondence should be addressed. E-mail: zhangkaicn@sina.com

word $X \in S$ which are either G or C. This constraint affects the thermodynamic properties of a word. Therefore, if all the words will assure similar GC content, all the DNA sequences must have similar thermodynamic characteristics.

1.2 Forbidden subsequences constraint

Due to different biochemical experimental requirement, each experiment must exclude undesirable forbidden subsequences. For any word $X \in S$, the word must not contain undesired subsequences all over the whole strand.

1.3 Hamming distance constraint

Hamming distance between two binary strings is the number of corresponding places where two characters differ. In DNA coding, Hamming distance is used to describe the non-similar degree between two DNA sequences, the greater Hamming distance, the less similar degree of two base pairs and less likely the mismatch hybridization.

The Hamming distance $H(X, Y)$ between two words X and Y is the number of positions where X differs from Y. Given k, for any word X, $Y \in S$, $H(X, Y) \geqslant k$. This constraint limits non-specific hybridizations between the Watson-Crick complements of some words X with a distinct word Y.

1.4 Slide mismatches constraint

If there are too many similar subsequences with a length long enough in word X, non-specific hybridizations between word X and its complement could take place. The slide mismatches constraint limits such a kind of non-specific hybridizations between word X and its complement. For any word $X \in S$, $H(X[i, \cdots, i+k], X[j, \cdots, j+k]) > 0$ for all i, j.

For instance, if X = CTAGATGCCTAGATCGA, there are two subsequences CTAGAT in X. When $k=6$, then $H(X[1, \cdots, 6], X[9, \cdots, 14]) = 0$. Such along subsequence could result in non-specific hybridizations in some biochemistry experiments.

2 Algorithm for DNA words design

Our DNA words design algorithm is based on a global heuristically search approach. It takes as input the desired strand length n, along with a specification of which constraints the set must satisfy, and attempts to find a set that meets these requirements. The algorithm performs a search in a complete space of DNA strand sets of fixed size that may violate the given constraints, using a heuristic search strategy.

First, we replace the nucleotides alphabet set {A, T, G, C} with quaternary alphabet set {0, 1, 2, 3}. Each nucleotide corresponds to one quaternary character from the set {0, 1, 2, 3}, for example, "0" means "C", "1" means "A", "2" means "T" and "3" means "G". So any DNA sequence with length n can be expressed as a quaternary with length n. All the quaternary forms an ordered set, which corresponds to the entire DNA sequences. There is a one to one mapping between each quaternary and each DNA sequence. When $n = 6$, the DNA sequences and the quaternary numbers are shown in Fig. 1.

000000 → CCCCCC
000001 → CCCCCA
000002 → CCCCCT
000003 → CCCCCG
000010 → CCCCAC
000011 → CCCCAA
…… ……
111111 → AAAAAA
…… ……
222222 → TTTTTT
…… ……
333333 → GGGGGG

Fig. 1. The DNA words and the corresponding quaternary, when $n=6$

Second, we designed a quaternary generator which can generate the quaternary number in proper order. When the algorithm generates a quaternary numeric code, all the required constraints need to test the code one by one. If the new word fulfills all the constraints, it is appended to the desired DNA sequences set. Then the numeric code will be added 1 to generate a new code, which also needs to be tested with the constraints. If the generated DNA code violated one or more constraints, the constraint will be turned into a carry number. If the carry is on the highest position, then the process terminates. Otherwise, the biggest carry numbers will be added to the DNA code to generate a new DNA numeric code which needs to be checked by the constraints.

The flow chart of the algorithm is described in Fig. 2.

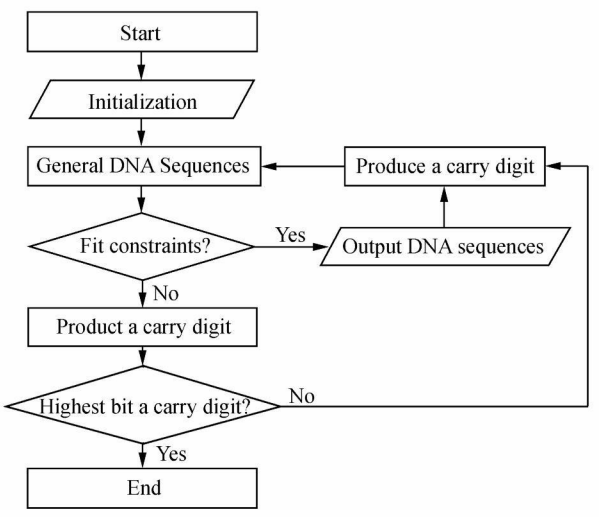


Fig. 2. The flow of DNA words design algorithm.

The algorithm begins with parameters initialization. Then it repeatedly generates a new sequence, and tests the sequence with the given constraints. The carry numbers are the parameters of the new sequence. The algorithm has been implemented on Pascal language complier of Borland Delphi 7. The source code has been made publicly available by our E-mails. Pseudo program code is given bellow:

```
program DNAWordGenerator var
  DNALength, HammingDistance : integer;
  GCcontent, SlideLength : integer;
  ForbiddenSequences : AnsiString;
begin
  Initial;
  repeat
    Generate a new DNA sequence;
    Analysis the new quaternary number;
    {
      GC content Constraint generate a carry number;
      Subsequence Constraint generate a carry number;
      Hamming distance Constraint generate carry number;
      Slide mismatch Constraint generate a carry number;
      (Additional Constraints)
    }
    If no carry number then
      Transform the quaternary number to DNA sequence;
      Append the DNA word to the sequences set S;
    Else
      Select the max carry number for new sequence;
  until Max carry > DNALength
end.
```

3 Convert constraints to carry numbers

This section shows how to convert the thermodynamic and combinatorial constraints to carry numbers. Other constraints can be added and be converted to a carry number in a similar way.

3.1 Convert GC content constraint

For example, the length of designed DNA words is 8, and the desired GC content should be 4. Let tested DNA sequence be "CGGCCCCC", then the corresponding quaternary number is "03300000". The number will be taken statistically from left to right to calculate the GC content. Fig. 3 shows how to convert the GC content constraint to a convergent condition.

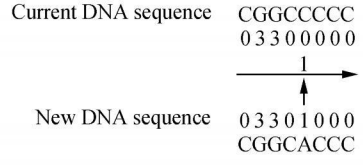
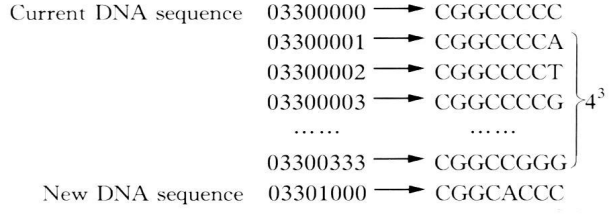


Fig. 3. Convert the GC content constraint to a carry number.

The algorithm statistics the G and C character from left to right. The word violates the GC content constraint, because the DNA word has 5 GC characters which is bigger than the desired GC content 4 when the algorithm statistics to position 4. The carry number at position 4 means that there are 4^3 quaternary codes which cannot satisfy the GC content constraint.



Because all of the above DNA sequences do not satisfy the GC content constraint, the carry will skip over 4^3 unnecessary DNA sequences constraints test. In this way, the algorithm can accelerate the convergent.

3.2 Convert forbidden subsequences constraint

The biochemistry experiment requests that the

designed DNA sequence must exclude some forbidden subsequence, such as "CCCCC", "AAAAA", "TTTTT", and "GGGGG". So these sequences will be checked with the given forbidden subsequences and if the sequence violates the constraint, a carry should be generated at the violate position. For example, the tested DNA sequence is "AAACCGGT", and "AAA" is a forbidden subsequence, then the corresponding quaternary number is "11100332". Fig. 4 shows how to convert the forbidden subsequences constraints to a convergent condition.

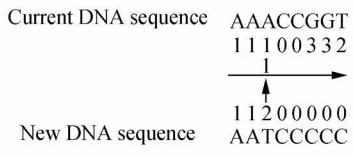
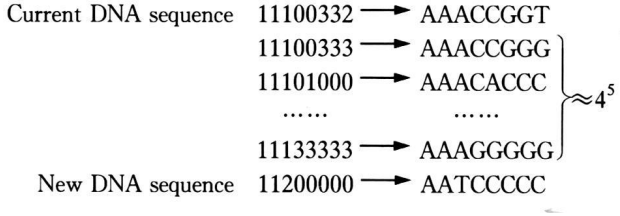


Fig. 4. Convert the forbidden subsequence constraint to a carry number.

The algorithm searches for the forbidden subsequence in the DNA word from left to right. A forbidden subsequence "AAA" was found, so the word violates the forbidden subsequences constraint. The carry number at position 6 means that there are 4^5 quaternary codes not satisfying the forbidden subsequences constraint.



Because all of the above DNA sequences do not satisfy the forbidden subsequences constraint, the carry will skip over 4^5 unnecessary DNA sequences constraints test.

3.3 Convert Hamming distance constraint

For example, Hamming distance of the designed DNA sequences must be equal to or greater than 4. If we have got a satisfied DNA sequence "CCACCAAT" whose corresponding number is "00100112", then we search the new DNA sequence from the quaternary number "00100113".

New DNA sequence will compare with the sequences "CCACCAAT" by Hamming distance. If the sequence violates the constraint, a carry should be generated at the violate position as Fig. 5 shows.

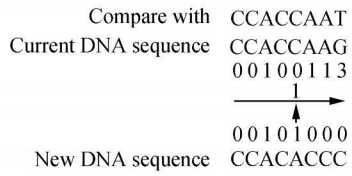
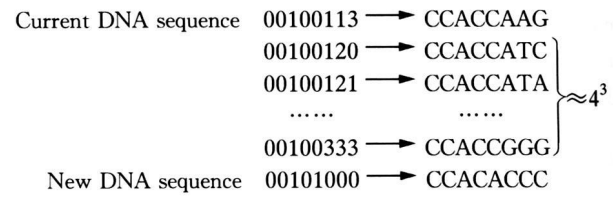


Fig. 5. Convert the Hamming distance constraint to a carry number.

The algorithm compares the current DNA word with satisfied words set and tests the word from left to right. There are 5 same characters in the DNA word, so the Hamming distance must be less than 4, and the word violates the Hamming distance constraint. In this example, the carry number at position 4 means that there are 4^3 quaternary codes not satisfying the Hamming distance constraint.

Because all of the above DNA sequences do not satisfy the Hamming distance constraint, the carry will skip over about 4^3 unnecessary DNA sequences constraints test.



3.4 Convert slide mismatches constraint

A DNA sequence must not contain two same subsequences with required length. If the sequence violates the constraint, a carry should be generated at the violate position. For example, the tested DNA sequence is "CTAGACTAGATCCGA", and the unique Watson-Crick complement is "GATCTGATCTAGGCT". While the sequence "CTAGACTAGATCCGA" has two same subsequences with length 5, which causes slide mismatches. Fig. 6 shows the slide mismatch hybridization and shows how to generate a carry number at the violate position.

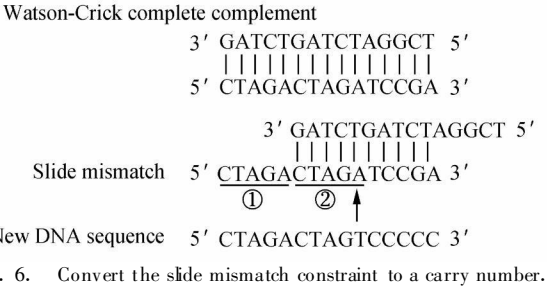
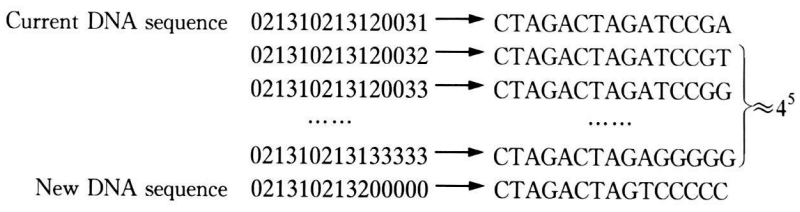


Fig. 6. Convert the slide mismatch constraint to a carry number.

The algorithm searchs for two same subsequences with given length from left to right. A subsequence "CTAGA" was found in the DNA word twice, so the word violates the slide mismatch constraint. The carry number at position 6 means that there are 4^5 quaternary codes not satisfying the slide mismatch constraint.



Because all of the above DNA sequences do not satisfy the slide mismatch constraint, the carry will skip over almost 4^5 unnecessary DNA sequences constraints test.

4 Conclusion

A new method of sequence design for DNA-based computation is introduced. Because the algorithm is based on global search, the DNA sequence set is one of the greatest sets which satisfy the constraints. The algorithm supports additional constraints, which can be integrated into our model in a straightforward way. In general, the more constraints are required, the more time will be consumed for most other DNA words design methods[4—9]. On the contrary, when more constraints are integrated, our algorithm may be fortunately less time-consuming, because additional constraints can generate a carry at higher position which skips over a large quantity of unnecessary DNA words test and accelerate the algorithm convergent.

References

1 Adleman LM. Molecular computation of solutions to combinatorial problems. Science, 1994, 266(11): 1021—1024
2 Lipton R. DNA solutions of hard computation problems. Science, 1995, 268(11): 542—545
3 Garey MR and Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: Freeman and Company, 1979
4 Frutos AG, Smith LM and Corn RM. Enzymatic ligation reactions of DNA "Word" on surfaces for DNA computing. Journal of the American Chemical Society, 1998, 120(40): 10277—10282
5 Frutos AG, Liu QH, Thiel AT, et al. Demonstration of a word design strategy for DNA computing on surface. Nucleic Acids Research, 1997, 25: 4748—4757
6 Feldkamp U, Banzhaf W and Rauhe H. A DNA sequence compiler. In: Proceedings of 6th DIMACS Workshop on DNA Based Computers, 2000
7 Deaton R, Garzon M, Murphy RC, et al. Genetic search of reliable encodings for DNA-based computation. In: 1ST Genetic Programming Conference, 1996
8 Deaton R, Murphy RC, Garzon M, et al. Good encodings for DNA-based solutions to combinatorial problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1999, 44: 247—258
9 Deaton R, Murphy RC, Rose JA, et al. A DNA based implementation of an evolutionary computation. In: Proceedings IEEE Conference on Evolutionary Computation, Indians, 1997, 267—271