Bioinformatics, 32(6), 2016, 828–834
doi: 10.1093/bioinformatics/btv678
Advance Access Publication Date: 20 November 2015
Original Paper

Sequence analysis

CDSfold: an algorithm for designing a protein-coding sequence with the most stable secondary structure

Goro Terai[1,2,*] Satoshi Kamegai[1,2] and Kiyoshi Asai[1,3,*]

[1]National Institute of Advanced Industrial Science and Technology (AIST), Koto-ku, Tokyo 135-0064, Japan, [2]INTEC Inc., Koto-ku, Tokyo 136-8637, Japan and [3]Graduate School of Frontier Sciences, University of Tokyo, Kashiwa 277-8562, Japan

*To whom correspondence should be addressed
Associate Editor: Ivo Hofacker

Received on April 23, 2015; revised on November 3, 2015; accepted on November 12, 2015

Abstract

Motivation: An important problem in synthetic biology is to design a nucleotide sequence of an mRNA that confers a desirable expression level of a target protein. The secondary structure of protein-coding sequences (CDSs) is one potential factor that could have both positive and negative effects on protein production. To elucidate the role of secondary structure in CDSs, algorithms for manipulating secondary structure should be developed.

Results: We developed an algorithm for designing a CDS with the most stable secondary structure among all possible ones translated into the same protein, and implemented it as the program CDSfold. The algorithm runs the Zuker algorithm under the constraint of a given amino acid sequence. The time and space complexity is $O(L^3)$ and $O(L^2)$, respectively, where L is the length of the CDS to be designed. Although our algorithm is slower than the original Zuker algorithm, it could design a relatively long (2.7-kb) CDS in approximately 1 h.

Availability and implementation: The CDSfold program is freely available for non-commercial users as stand-alone and web-based software from http://cdsfold.trahed.jp/cdsfold/.

Contacts: terai-goro@aist.go.jp or asai@k.u-tokyo.ac.jp

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Protein expression levels are reportedly affected by various types of mRNA features, such as codon usage (Sharp and Li, 1987), rare codon clusters (Kane, 1995), secondary structures in untranslated regions (Mignone *et al.*, 2002) and protein-coding sequences (CDSs; see below), Kozak motif (Kozak, 1984) and Shine-Dalgarno sequence (Salis *et al.*, 2009). Among them, secondary structure is difficult to handle because of the need to consider base pair interactions between distant nucleotides and complex energy parameters assigned to each part of the secondary structure. This may be one of the reasons that some recent methods for designing CDSs do not consider secondary structural features (Chin *et al.*, 2014; Chung and Lee, 2012; Gaspar *et al.*, 2012; Raab *et al.*, 2010). The secondary structure of CDSs, however, can have substantial effects on protein production as described below, although the mechanisms by which secondary structure affects protein production are not understood fully, especially in eukaryotes.

Previous studies have found that secondary structures near the start codon tend to be unstable in both prokaryotes and eukaryotes (Gu *et al.*, 2010; Kertesz *et al.*, 2010; Robbins-Pianka *et al.*, 2010; Wan *et al.*, 2014), suggesting that secondary structures have negative effects on protein expression levels. Indeed, in prokaryotes, it was shown that a stable secondary structure around the start codon reduced protein expression levels, possibly by blocking the access of the ribosome to the ribosome binding site (Kudla *et al.*, 2009). Also, in eukaryotes, a previous study demonstrated that the secondary

structure around the start codon strongly reduced protein expression levels (Vega Laso *et al.*, 1993). This was confirmed by a recent study based on the large-scale measurements of protein abundance for over 2000 sequence variants in yeast (Dvir *et al.*, 2013). Therefore, it may be possible to decrease protein expression levels by artificially inserting a stable secondary structure near the start codon in both prokaryotes and eukaryotes.

Secondary structures far downstream of the start codon may also have effects on protein production. It is commonly believed that the secondary structures in CDSs impede translation elongation, resulting in a reduction of protein production. It has, however, been pointed out that slow elongation may not result in decreased protein production as long as there is sufficient space between elongating ribosomes (Plotkin and Kudla, 2011). Moreover, it has been suggested that slow elongation can assist in proper co-translational protein folding (Kramer *et al.*, 2009). Therefore, slow elongation does not necessarily decrease protein expression levels. Moreover, it cannot be ruled out that the secondary structures in CDSs have positive, rather than negative, effects on protein production by protecting mRNAs from various types of negative factors such as RNase digestion, down-regulation by microRNAs, and unintended interactions with other mRNA molecules. Interestingly, recent studies based on genome-wide measurements of RNA secondary structure in yeast (Kertesz *et al.*, 2010), found a high correlation between secondary structure and protein expression levels (Park *et al.*, 2013; Zur and Tuller, 2012); mRNAs of strongly expressed genes tend to form a more stable secondary structure. Although still arguable, these studies suggest a causal relationship between protein expression levels and the stability of the secondary structures of CDSs. Therefore, it is worth investigating whether (and to what degree) artificially stabilized secondary structures in CDSs decrease or enhance protein expression levels.

To elucidate the effect of secondary structures in CDSs as well as utilize the features of secondary structures to control protein expression levels, algorithms for handling the secondary structures of CDSs should be developed. Currently, however, there are very few tools for stabilizing the secondary structures of CDSs. Gaspar et al. recently developed the mRNA optimiser (Gaspar *et al.*, 2013), which is based on a simulated annealing algorithm. Although the method was intended to remove secondary structures from CDSs, it can also be used to stabilize the secondary structures. Here, we developed a different approach that was based on a dynamic programming algorithm. It designs the CDS with the most stable secondary structure among all possible ones translated into the same protein. In our algorithm, the Zuker algorithm (Zuker and Stiegler, 1981) is executed under the constraint of a given amino acid sequence. Although our algorithm uses larger DP matrices than the original Zuker algorithm, the fundamental procedure of both algorithms is the same. Therefore, as with the Zuker algorithm, the time and space complexity of our algorithm is $O(L^3)$ and $O(L^2)$, respectively, where L is the length of the CDS to be designed.

2 Methods

We start with a description of the simplified version of our algorithm, which designs a CDS with the maximum number of base pairs. The algorithm executes the Nussinov algorithm (Nussinov *et al.*, 1978) under the constraint of a given amino acid sequence. The original Nussinov algorithm is as follows:

2.1 Nussinov algorithm

We are given a nucleotide sequence. Let L be the length of this sequence. Among the numerous secondary structures that can be formed for the nucleotide sequence, the maximally base-paired structure can be obtained by the following algorithm:

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1) + \delta\langle i,j\rangle \\ \max\limits_{i<k<j} [\gamma(i,k) + \gamma(k+1,j)] \end{cases} \quad (1)$$

where $\gamma(i,j)$ is the maximum number of base pairs that can be formed for a sub-sequence from position i to j, and $\delta\langle i,j\rangle$ takes 1 if the nucleotides in positions i and j can form a base pair, or 0 otherwise.

The diagonal elements of the γ matrix and their neighboring elements, $\gamma(i,i-1)$, are initialized to be 0. Then, the Nussinov algorithm recursively calculates $\gamma(i,j)$ until $\gamma(1,L)$ is obtained. The value of $\gamma(1,L)$ is the number of base pairs in the maximally base-paired structure.

The positions of the base-paired nucleotides in the maximally base-paired secondary structure can be obtained by tracing back the γ matrix, beginning from $\gamma(1,L)$ (see Durbin *et al.* (1998) for a concise explanation).

2.2 Nussinov algorithm under amino acid constraints

We are given an amino acid sequence. Let L be the length of the CDS to be designed. Let N_i be the set of possible nucleotides at the i th position of the CDS. Let n_i be the realization of a nucleotide at position i. The size of N_i varies from 1 to 4 depending on the amino acid residue and its codon position. Specifically, 1 or 2 nucleotides are allowed in the first and second codon positions of all amino acid residues, whereas 2 or 4 nucleotides are allowed in the third position in most cases.

For some amino acid residues, there is a 'dependency' in their codons. For example, serine is coded for by 6 codons: AGU, AGC, UCU, UCC, UCA and UCG. If the first codon position of serine is A, then the second position must be G. To cope with such dependencies, let $N_i|n$ be the set of allowable nucleotides at position i *after* nucleotide n. Similarly, let $N_i \wedge n$ be the set of allowable nucleotides at position i *before* nucleotide n.

For leucine and arginine, there are nucleotide dependencies between the *first* and *third* codon positions. As described later in this section, we can convert the non-adjacent nucleotide dependencies in leucine and arginine into adjacent ones. Here, we temporarily assume that there are no such nucleotide dependencies in leucine and arginine.

The algorithm for designing a CDS with the maximum number of base pairs is as follows:

Algorithm 1: the Nussinov algorithm under amino acid constraints, fill stage

$$\gamma^{n_i,n_j}(i,j) = \max \begin{cases} \max\limits_{n_{i+1}\in N_{i+1}|n_i} [\gamma^{n_{i+1},n_j}(i+1,j)] \\ \max\limits_{n_{j-1}\in N_{j-1}\wedge n_j} [\gamma^{n_i,n_{j-1}}(i,j-1)] \\ \max\limits_{n_{i+1}\in N_{i+1}|n_i,\, n_{j-1}\in N_{j-1}\wedge n_j} [\gamma^{n_{i+1},n_{j-1}}(i+1,j-1) + \delta'\langle n_i,n_j\rangle] \\ \max\limits_{i<k<j,\, n_k\in N_k,\, n_{k+1}\in N_{k+1}|n_k} [\gamma^{n_i,n_k}(i,k) + \gamma^{n_{k+1},n_j}(k+1,j)] \end{cases} \quad (2)$$

where $\gamma^{n_i,n_j}(i,j)$ retains the maximum number of base pairs among the maximally base-paired secondary structures of the sub-sequences

in which the first and last nucleotide is n_i and n_j, respectively, and $\delta'\langle n_i, n_j\rangle$ takes 1 if n_i and n_j can form a base pair, or 0 otherwise.

Similar to the original Nussinov algorithm, the diagonal elements and their neighboring elements, $\gamma^{n_i,n_{i-1}}(i, i-1)$, are initialized to be 0 for all allowable n_i and n_{i-1}. In addition, $\gamma^{n_i,n_j}(i,j)$ elements for all unallowable n_i and n_j under amino acid constraints are initialized and fixed to be -∞.

Then, Algorithm 1 recursively calculates $\gamma^{n_i,n_j}(i,j)$ until $\gamma^{n_1,n_L}(1, L)$ is obtained for all possible n_1 and n_L. The maximum value of $\gamma^{n_1,n_L}(1,L)$ over all possible n_1 and n_L is the maximum number of base pairs among the maximally base-paired secondary structures of all possible CDSs translated into the given amino acid sequence.

Note that multiple γ elements in each i, j position, which are indexed by n_i and n_j, are not needed for the amino acid residues other than serine, leucine and arginine. If a given amino acid sequence does not contain serine, leucine, or arginine (in other words, all codon positions are independent), a single γ element in each i, j is sufficient to calculate the maximum number of base pairs over all possible CDSs. The use of multiple γ elements, however, is relevant when we extend Algorithm 1 to energy minimization based on the energy model obtained by the Turner group (Mathews *et al.*, 1999), in which various types of energetic contributions are determined according to the adjacent nucleotides.

2.2.1 Obtaining the nucleotide sequence of a CDS by backtracking

In the original Nussinov algorithm, the maximally based-paired secondary structure can be obtained by tracing back the γ matrix. In the Nussinov algorithm under amino acid constraints, the nucleotide sequence of a CDS with the maximum number of base pairs, as well as its maximally based-paired secondary structure, can be obtained by a similar backtracking algorithm. Let $\gamma^{\hat{n}_1,\hat{n}_L}(1,L)$ be the maximum value of $\gamma^{n_1,n_L}(1,L)$ for all possible n_1 and n_L. The pseudocode of the backtracking algorithm is described as Algorithm 2.

Algorithm 2

```
1:  Initialization:
2:    Push(1, L, n̂1, n̂L) onto stack.
3:    cds(1) ← n̂1    ▷ Record n̂1 and n̂L as the first and
4:    cds(L) ← n̂L    ▷ last nucleotide, respectively.
5:  while stack is not empty do
6:    pop(i, j, ni, nj)
7:    if i ≥ j then
8:      continue;
9:    for all ni+1 ∈ Ni+1|ni do
10:     if γ^{ni,nj}(i, j) = γ^{ni+1,nj}(i + 1, j) then
11:       cds(i + 1) ← ni+1
12:       push (i + 1, j, ni+1, nj)
13:       continue;
14:   for all nj−1 ∈ Nj−1 ∧ nj do
15:     if γ^{ni,nj}(i, j) = γ^{ni,nj−1}(i, j − 1) then
16:       cds(j − 1) ← nj−1
17:       push (i, j − 1, ni, nj−1)
18:       continue;
19:   for all ni+1 ∈ Ni+1|ni and nj−1 ∈ Nj−1 ∧ nj do
20:     if γ^{ni,nj}(i, j) =
21:         γ^{ni+1,nj−1}(i + 1, j − 1) + δ′⟨ni, nj⟩ then
22:       cds(i + 1) ← ni+1
23:       cds(j − 1) ← nj−1
24:       if δ′⟨ni, nj⟩ = 1 then
25:         Record i, j as base-paired position
26:       push (i + 1, j − 1, ni+1, nj−1)
27:       continue;
28:   for all k such that i < k < j do
29:     for all nk ∈ Nk and nk+1 ∈ Nk+1|nk do
30:       if γ^{ni,nj}(i, j) =
31:         γ^{ni,nk}(i, k) + γ^{nk+1,nj}(k + 1, j) then
32:         cds(k) ← nk
33:         cds(k + 1) ← nk+1
34:         push (i, k, ni, nk)
35:         push (k + 1, j, nk+1, nj)
36:         continue;
```

Starting from $\gamma^{\hat{n}_1,\hat{n}_L}(1,L)$ element, the algorithm finds 1 or 2 elements from which $\gamma^{\hat{n}_1,\hat{n}_L}(1,L)$ is *derived*. We use the word *derived* when the equality in line 10, 15, 20, or 30 holds. For example, if the equality in line 10 holds, $\gamma^{n_i,n_j}(i,j)$ is derived from $\gamma^{n_{i+1},n_j}(i+1,j)$. Then, the algorithm recursively finds 1 or 2 elements from which each of the newly found elements is derived. As 2 elements at most are derived in each recursion (see line 30), the stack data structure is used to store them.

The nucleotide sequence of a CDS is obtained by recording 1 or 2 nucleotides in each recursion. For example, in line 10, if $\gamma^{n_i,n_j}(i,j)$ is derived from $\gamma^{n_{i+1},n_j}(i+1,j)$, the algorithm records n_{i+1} as the $i+1$th nucleotide in line 11. The position of a base pair in the maximally base-paired secondary structure is obtained by recording base-paired positions in line 25 if $\delta'\langle n_i, n_j\rangle = 1$ and $\gamma^{n_i,n_j}(i,j)$ is derived from $\gamma^{n_{i+1},n_{j-1}}(i+1,j-1)$ in line 20.

2.2.2 Introducing extended nucleotides

For leucine and arginine, there are nucleotide dependencies between the *first* and *third* codon positions. For example, leucine is coded for by 6 codons: UUA, UUG, CUU, CUC, CUA and CUG. If the first position is U, the third position is restricted to A or G. Algorithms 1 and 2 can cope with the dependency between adjacent nucleotides, but cannot consider the dependencies between the first and third positions. Fortunately, by introducing extended nucleotides, we can convert the non-adjacent nucleotide dependencies in leucine and arginine into adjacent ones. For leucine, we classified the nucleotide U in the second codon position into 2 different Us based on the nucleotide in the third position (Fig. 1). The second position is denoted by U^{AG} and U^{CU} if the third position is G or A and C or U, respectively. If the first position is C, then the second position can be both U^{AG} and U^{CU}. The extended nucleotides we used here and their dependencies are summarized in Figure 1.

2.3 Zuker algorithm under amino acid constraints

In the Zuker algorithm (Zuker and Stiegler, 1981), the most stable secondary structure in terms of the Turner energy model can be calculated for a given nucleotide sequence. Although the Zuker algorithm is more complex than the Nussinov algorithm, the

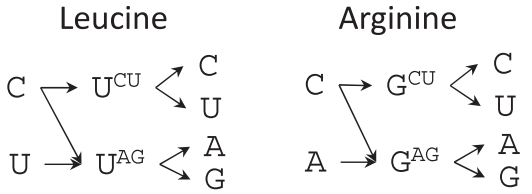

Fig. 1. Extended nucleotide representations for the second codon position of leucine and arginine. Arrows indicate the dependencies of the adjacent nucleotides. For example, U^{CU} in leucine can be followed by C or U

fundamental procedure of both algorithms is the same; namely, the optimal secondary structure is obtained from that of shorter sub-sequences. We extended the Nussinov algorithm under amino acid constraints so it can execute the Zuker algorithm under amino acid constraints. It is the Zuker version of the algorithm that was implemented in the CDSfold program.

Algorithm 3: the Zuker algorithm under amino acid constraints, fill stage

$$F^{n_i,n_j}(i,j) = \min \begin{cases} \min\limits_{n_{i+1}\in N_{i+1}|n_i} [F^{n_{i+1},n_j}(i+1,j)] \\ \min\limits_{n_{j-1}\in N_{j-1}\wedge n_j} [F^{n_i,n_{j-1}}(i,j-1)] \\ C^{n_i,n_j}(i,j) \\ \min\limits_{i<k<j,\, n_k\in N_k,\, n_{k+1}\in N_{k+1}|n_k} [F^{n_i,n_k}(i,k) + F^{n_{k+1},n_j}(k+1,j)] \end{cases} \quad (3)$$

where $F^{n_i,n_j}(i,j)$ retains the smallest minimum free energy (MFE) over a set of sub-sequences in which the first and last nucleotide is n_i and n_j, respectively. The recursion equations of Algorithm 3 are almost the same as those of Algorithm 1; the only difference is that the third equation is replaced by $C^{n_i,n_j}(i,j)$, which retains the smallest MFE provided that n_i and n_j form a base pair.

When calculating $C^{n_i,n_j}(i,j)$, we must consider the following 5 cases: nucleotides n_i and n_j form a base pair that

1. immediately follows another base pair,
2. encloses a hairpin loop,
3. encloses a bulge loop,
4. encloses internal loops,
5. encloses a multi-loop.

2.3.1 Case 1: Two consecutive base pairs

In the first case, $C^{n_i,n_j}(i,j)$ is obtained by considering the energy of 2 consecutive base pairs; this energy is commonly called *stacking energy*. For a particular base pair between n_i and n_j, we must consider all possible base pairs between N_{i+1} and N_{j-1}. Therefore, the smallest MFE when a base pair n_i and n_j immediately follows another base pair is obtained as follows:

$$\min_{\substack{n_{i+1}\in N_{i+1}|n_i \\ n_{j-1}\in N_{j-1}\wedge n_j}} \left[C^{n_{i+1},n_{j-1}}(i+1,j-1) + E_{\text{stacking}}\langle n_i, n_j, n_{i+1}, n_{j-1}\rangle \right] \quad (4)$$

where E_{stacking} is a function that calculates the stacking energy depending on a base pair between n_i and n_j that immediately follows a base pair between n_{i+1} and n_{j-1}. For example, a G:C base pair that follows C:G is far more stable (−3.4 kcal/mol) than A:U following U:A (−0.9 kcal/mol). The stacking energy is assumed to be $+\infty$ if a base pair cannot be formed either between n_i and n_j or between n_{i+1} and n_{j-1}.

2.3.2 Case 2: Hairpin loop

In the second case, nucleotides n_i and n_j enclose a hairpin loop. An example of a hairpin loop is shown in Figure 2a. In the Turner energy model, the energy of a hairpin loop depends on the nucleotides in the 4 positions indicated in Figure 2a. The smallest MFE when n_i and n_j enclose a hairpin loop is as follows:

$$\min_{\substack{n_{i+1}\in N_{i+1}|n_i \\ n_{j-1}\in N_{j-1}\wedge n_j}} \left[E_{\text{hairpin}}\langle n_i, n_j, n_{i+1}, n_{j-1}, v^h\rangle \right] \quad (5)$$

where E_{hairpin} is a function that calculates the energy of a hairpin loop depending on the 4 nucleotides and the length of the hairpin loop $v^h (= j-i-1)$. The energy of the hairpin loop is assumed to be $+\infty$ if a base pair cannot be formed between n_i and n_j.

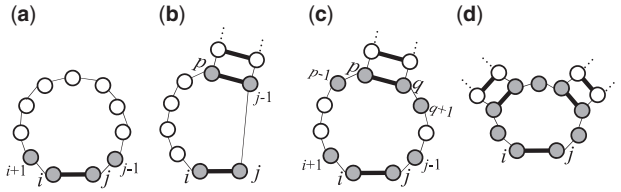

Fig. 2. Nucleotides affecting the energy of 4 types of loop structures. (a) Hairpin loop, (b) bulge loop, (c) internal loops and (d) multi-loop. A circle represents a nucleotide. A base pair is denoted by the 2 nucleotides connected with a bold line. Filled circles are the nucleotides that affect the energy of the loops. Letters associated with the circles indicate positions on a nucleotide sequence

In the Turner energy model, there is a special table specifying the energy parameters for small hairpin loops of sizes 5, 6 and 8-nt (including a closing base-pair). For those hairpins, all nucleotides in the hairpin loop influence the free energy. To take this into account, we enumerate all possible 5, 6, 8-mers starting from each position of a CDS before running the recursion, and check whether there can be the special hairpin loops between position i and j when $j-i+1 = 5$, 6 or 8.

2.3.3 Case 3: Bulge loop

In the third case, n_i and n_j enclose a bulge loop. Figure 2b is an example of a bulge loop that is located between position i and p. The smallest MFE when n_i and n_j enclose a bulge loop is:

$$\min_{\substack{i+1<p<j-1 \\ n_p\in N_p \\ n_{j-1}\in N_{j-1}\wedge n_j}} \left[C^{n_p,n_{j-1}}(p,j-1) + E_{\text{bulge}}\langle n_i, n_p, n_{j-1}, n_j, v^l\rangle \right] \quad (6)$$

where E_{bulge} is a function that calculates the energy of a bulge loop depending on the 4 nucleotides indicated in Figure 2b and the length of the bulge loop $v^l (= p-i-1)$. The energy of the bulge loop is assumed to be $+\infty$ if a base pair cannot be formed either between n_i and n_j or n_p and n_{j-1}.

The smallest MFE when a bulge loop is located on the right side of the stem can be calculated similarly (see Supplementary Methods).

2.3.4 Case 4: Internal loops

In the fourth case, n_i and n_j enclose internal loops (Fig. 2c). In the Turner energy model, the energy of the internal loops depends on the 8 positions indicated in Figure 2c. The smallest MFE when a base pair formed by n_i and n_j encloses internal loops is as follows:

$$\min_{\substack{i+1<p<q<j-1 \\ n_p\in N_p, n_q\in N_q \\ \psi\in\Psi(n_i,n_p,n_q,n_j)}} \left[C^{n_p,n_q}(p,q) + E_{\text{intloop}}\langle \psi, v^l, v^r\rangle \right] \quad (7)$$

where $\Psi(n_i,n_p,n_q,n_j)$ is a set of possible nucleotides in the 8 positions, provided that n_i, n_j, n_p and n_q are given. E_{intloop} is a function that calculates the energy of the internal loops depending on ψ, as well as v^l and v^r, which are the length of the left and right side of the internal loops, respectively. The energy of the internal loops is assumed to be $+\infty$ if a base pair cannot be formed either between n_i and n_j or n_p and n_q.

2.3.5 Case 5: Multi-loop

In the last case, n_i and n_j enclose a multi-loop. A multi-loop is a loop structure enclosed by 3 or more base pairs. An example of a multi-loop enclosed by 3 base pairs is shown in Figure 2d. The smallest MFE when n_i and n_j enclose a multi-loop is as follows:

$$\min_{\substack{n_{i+1} \in N_{i+1} | n_i \\ n_{j-1} \in N_{j-1} \wedge n_j}} \left[M^{n_{i+1}, n_{j-1}}(i+1, j-1) + ML_{\text{closing}}\langle n_i, n_j \rangle \right] \quad (8)$$

where $M^{n_{i+1}, n_{j-1}}(i+1, j-1)$ is the smallest MFE of sub-sequences from n_{i+1} to n_{j-1} that are a part of a multi-loop. ML_{closing} is the energetic penalty imposed when a base pair between n_i and n_j encloses a multi-loop. The calculation of $M^{n_{i+1}, n_{j-1}}$ is slightly complicated and involves yet another DP matrix (for details, see Supplementary Methods).

Once all values in the DP matrices are calculated, a traceback algorithm similar to Algorithm 2 can be used to obtain the nucleotide sequence of a CDS with the most stable secondary structure. For conciseness, the traceback algorithm and the complete details of Algorithm 3 are described in the Supplementary Methods.

In Algorithm 3, the energy of dangling ends is not taken into account, because considering it makes our algorithm much more complicated and slower. Therefore, the current version of CDSfold does not consider dangling energy. However, we show the algorithm that can consider dangling energy in the Supplementary Methods.

2.4 Comparison with simulated annealing

As a baseline method for comparison, we consider a more straightforward approach based on the simulated annealing algorithm. It iteratively optimizes the secondary structure of a CDS by successive mutations under amino acid constraints. The procedure is as follows:

1. An initial CDS is generated randomly that is used as a *template*. In addition, the number of iterations, denoted by i, is set to 0.
2. The MFE of the template is calculated in kcal/mol (denoted by e).
3. A single nucleotide in the template is randomly mutated under the amino acid constraints. i is increased by 1.
4. The MFE of the mutated CDS is calculated in kcal/mol (denoted by e').
5. If $e \geq e'$, the template is updated to the mutated CDS and e is also updated to e'. Otherwise, the template and e are updated with the following probability:

$$P_{\text{update}} = \exp\left(\frac{e - e'}{cT^i}\right) \quad (9)$$

where T is a temperature parameter that takes $0 < T < 1$. The T^i value (T with exponent i) approaches 0 as the number of iterations i increases, which means that P_{update} is also close to 0 as i increases. Thus, the larger i becomes, the more unlikely the template is updated. c is another parameter that is set to $c = 1000$ in this study to make P_{update} close to 1 when i is small.

Processes 3–5 are repeated until a predefined criterion is satisfied. In this study, we terminated the algorithm if the MFE (e value) is the same during 1000 successive iterations or the running time exceeds 100 h. For the calculation of MFE, we used the RNAfold program in Vienna RNA Package version 2.1.1 (Lorenz *et al.*, 2011). We used RNAfold with the -d 0 option to use the 'no-dangle model'. The difference between our simulated annealing approach and mRNA optimiser is an objective function to be optimized. The mRNA optimiser uses a 'pseudo-MFE' as an objective function of simulated annealing, which is proposed to have a good correlation with the real MFE (Gaspar *et al.*, 2013).

2.5 Computational environment

All computational experiments were performed on the Chimera cluster system at AIST, which had 176 Intel Xeon E5550 CPUs (2.53 GHz).

3 Results and discussion

3.1 Running time for real biological sequences

The running time of our algorithm is affected by the amino acid composition of the input sequence, as well as the length of the CDS to be designed. If an input sequence contains more residues coded for by more codons, our algorithm spends more time assessing all nucleotide combinations that affect the energy of the secondary structure.

To estimate empirically the influence of different amino acid compositions and evaluate the applicability of our method to real biological data, we measured the running time of CDSfold using proteins in the UniProt database (UniProt Consertium, 2015). We obtained amino acid sequence segments of the same length from the start codon of 100 randomly selected proteins. Here, we obtained sequence segments with lengths of 100, 300, 500, 700, 900, 1100, 1300 and 1500 amino acids, separately. Figure 3 shows the mean and standard deviation (SD) of the running time for each length. On average, it takes less than 1 h to design a CDS of length 2.7 kbp. As more than 95% of the sequences in the UniProt database are shorter than this length, our algorithm can design a CDS within 1 h for the majority of proteins. The figure also shows the SD of the running time is approximately 10–20% of the mean, indicating that amino acid composition has a modest effect on running time.

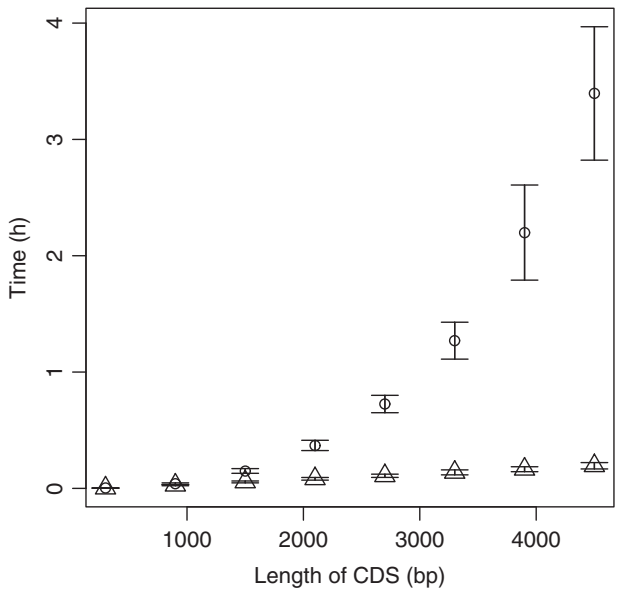

Fig. 3. Running time of CDSfold. Circles represent mean running times. Triangles represent mean running times when $W = 500$ bp. The error bars represent standard deviation

If an exceptionally long CDS must be designed, we can reduce the running time by limiting the maximum distance between base-paired nucleotides (denoted by W). If W is limited, time complexity becomes $O(W^2L)$, indicating that the running time increases only linearly on L. Figure 3 also shows the running time when $W = 500$ bp.

3.2 Comparison with simulated annealing

As a baseline method for comparison, we developed a more straightforward approach based on simulated annealing (see Section 2). We randomly selected 100 human proteins from the UniProt database (UniProt Consortium, 2015) and stabilized the secondary structure of their CDSs by both CDSfold and simulated annealing. The temperature parameter T of simulated annealing was set to 0.99. The MFE obtained by CDSfold was lower than that of simulated annealing, as expected (Fig. 4). The difference of the MFE becomes larger as the length of the CDS becomes longer (note that long CDSs generally have lower MFEs, and see also Supplementary Figure S1 that shows the MFE of even longer CDSs). It should also be noted that the running time of CDSfold was much shorter than that of simulated annealing; in most cases, CDSfold was 10–100 times faster than simulated annealing. Table S1 shows a list of the 100 genes examined and the running time for each gene.

The optimization result of simulated annealing may vary depending on the temperature parameter T. However, when we used different parameter values ($T = 0.9$ and 0.999), the same overall tendency was observed, although a slightly improved MFE of short CDSs was seen with $T = 0.999$ (Supplementary Fig. S2). Therefore, using straightforward simulated annealing, it may be difficult to obtain the optimum solution, especially for long CDSs.

3.3 Software

CDSfold is available as standalone software written in C++ as well as a web-based application. As an example, we designed the CDS of enhanced green fluorescent protein, EGFP (GenBank KM042177.1), which is often used to evaluate protein expression levels. Figure 5 shows the result page when the maximum distance between base-paired nucleotides W is limited to 50 bp. Under this limitation, the designed CDS forms the most stable secondary structure among all possible CDSs translated into the same EGFP. The MFE of the optimized CDS sequence is lower (−363.7 kcal/mol) than that of the original CDS obtained from GenBank (−238.90 kcal/mol).

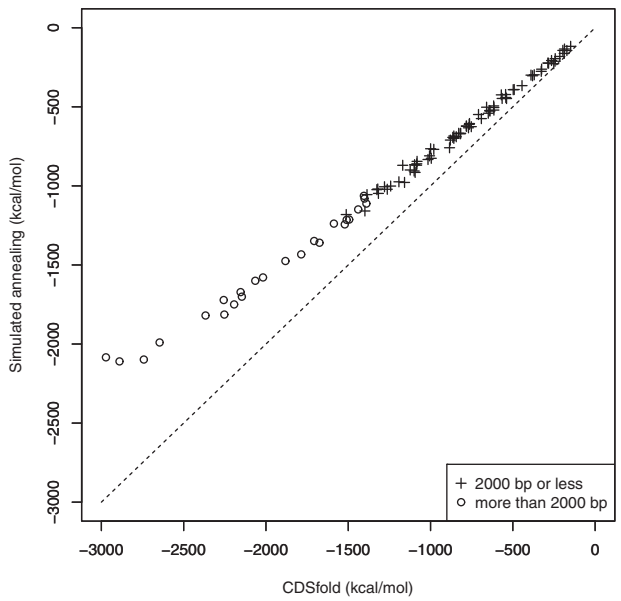

Fig. 4. Comparison of the MFE of CDSs designed by CDSfold and a simulated annealing algorithm

3.4 Practical use of CDSfold

3.4.1 Avoidance of certain codons

In practical gene design, rare codons are often avoided because they are suspected to reduce translation elongation. Our software can be used to design an optimal CDS without rare codons. Users can indicate a set of codons that should be avoided.

3.4.2 CDSfold sequence as a starting point for heuristic optimizations

A CDS designed by CDSfold may be effectively used as a starting point for heuristic optimizations. As an example, we designed a CDS having high Codon Adaptation Index (CAI) (Sharp and Li, 1987) as well as stable secondary structure, using simulated annealing described in the Section 2. For this purpose, we defined a new energy score in which both CAI and MFE are taken into account.

$$e = \mathrm{MFE} * \mathrm{CAI}^{\lambda} \tag{10}$$

where λ is a weight parameter associated with CAI. The larger λ is, the greater the contribution of CAI is. Supplementary Figure S3 shows the optimization results of a CDS encoding EGFP. By using various λ values, we could obtain a set of CDSs having different CAI and MFE. It is interesting that, when the CDSfold sequence was used as a starting point, we could obtain better CDSs than when a randomly generated CDS was used as a starting point. In this example, CAI was calculated using top 1000 highly expressed genes in *S. cerevisiae* (Holstege *et al.*, 1998) as a reference gene set.

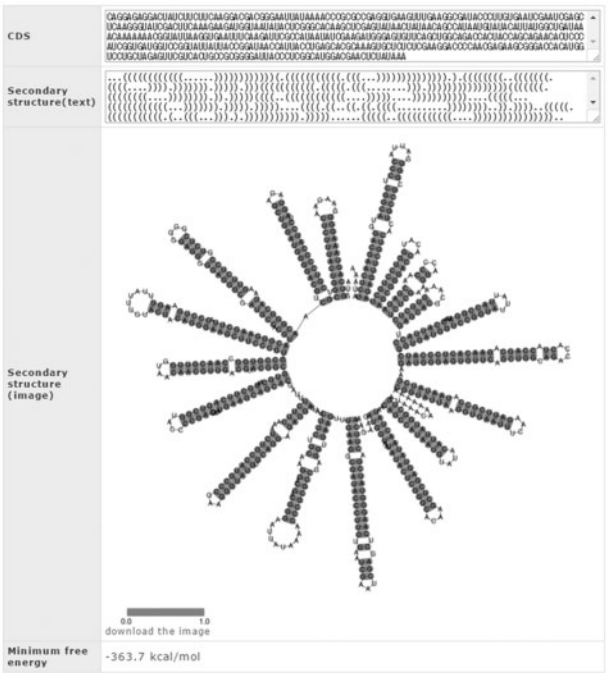

Fig. 5. An example of the result page of our web-based application. The nucleotide sequence of an optimized CDS, predicted secondary structure and MFE are displayed

3.4.3 Design of (partially) unstable structure

To evaluate the effect of secondary structure, it is important to design unstable secondary structures as well as stable ones. In this study, we developed a fast heuristic approach for designing a CDS with unstable structure, based on a pseudo-MFE similar to the one developed by Gaspar *et al.* (2012) (see Supplementary Methods). The web-application has radio buttons for switching between energy minimization by the dynamic programing and energy maximization by the heuristic approach.

The effect of the secondary structure may vary depending on the position on a CDS. Therefore, we developed a two-step procedure for designing a CDS with partially unstable secondary structure and implemented it in CDSfold. In the first step, energy minimization by the dynamic programing is performed under the assumption that nucleotides in a given interval do not form base-pairs. Specifically, we set $C^{n_i,n_j}(i,j)$ in Eq. 3 to be $+\infty$ when i and/or j is within the interval. Note that a CDS designed under this assumption can have base-pairs in the interval when this assumption is removed. Therefore, in the second step, the heuristic pseudo-MFE maximization is performed on the interval to remove secondary structure. Users can indicate an interval where the secondary structure should be unstable.

4 Conclusion

We developed an algorithm for designing a CDS with the most stable secondary structure in terms of the Turner energy model. It was shown that CDSfold can design CDSs of most genes within 1 h. It was also shown that our algorithm can design CDSs with a more stable secondary structure than the method based on simulated annealing, especially when a CDS is long. The algorithm is useful for investigating the effect of the secondary structure of a CDS as well as utilizing features of the secondary structure to control protein expression levels. Note that, by changing the secondary structure of a CDS, potential binding sites for other regulatory proteins or factors might be also changed, which might be responsible for the change in expression level. Therefore, the effect of the secondary structure should be investigated using at least several different proteins.

Acknowledgements

We thank the members of the Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology for useful discussions.

Funding

This work was supported by the commission for the Development of Artificial Gene Synthesis Technology for Creating Innovative Biomaterial from the Ministry of Economy, Trade and Industry, Japan.

Conflict of Interest: none declared.

References

Chin,J.X. *et al.* (2014) Codon Optimization OnLine (COOL): a web-based multi-objective optimization platform for synthetic gene design. *Bioinformatics*, **30**, 2210–2212.

Chung,B.K. and Lee,D.Y. (2012) Computational codon optimization of synthetic gene for protein expression. *BMC Syst. Biol.*, **6**, 134.

Durbin,R. *et al.* (1998) *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, chapter RNA structure analysis. Cambridge University Press, Cambridge, UK, pp. 260–298.

Dvir,S. *et al.* (2013) Deciphering the rules by which 5'-UTR sequences affect protein expression in yeast. *Proc. Natl. Acad. Sci. U. S. A.*, **110**, E2792–2801.

Gaspar,P. *et al.* (2012) EuGene: maximizing synthetic gene design for heterologous expression. *Bioinformatics*, **28**, 2683–2684.

Gaspar,P. *et al.* (2013) mRNA secondary structure optimization using a correlated stem-loop prediction. *Nucleic Acids Res.*, **41**, e73.

Gu,W. *et al.* (2010) A universal trend of reduced mRNA stability near the translation-initiation site in prokaryotes and eukaryotes. *PLoS Comput. Biol.*, **6**, e1000664.

Holstege,F.C. *et al.* (1998) Dissecting the regulatory circuitry of a eukaryotic genome. *Cell*, **95**, 717–728.

Kane,J.F. (1995) Effects of rare codon clusters on high-level expression of heterologous proteins in *Escherichia coli*. *Curr. Opin. Biotechnol.*, **6**, 494–500.

Kertesz,M. *et al.* (2010) Genome-wide measurement of RNA secondary structure in yeast. *Nature*, **467**, 103–107.

Kozak,M. (1984) Point mutations close to the AUG initiator codon affect the efficiency of translation of rat preproinsulin in vivo. *Nature*, **308**, 241–246.

Kramer,G. *et al.* (2009) The ribosome as a platform for co-translational processing, folding and targeting of newly synthesized proteins. *Nat. Struct. Mol. Biol.*, **16**, 589–597.

Kudla,G. *et al.* (2009) Coding-sequence determinants of gene expression in *Escherichia coli*. *Science*, **324**, 255–258.

Lorenz,R. *et al.* (2011) ViennaRNA Package 2.0. *Algorithms Mol. Biol.*, **6**, 26.

Mathews,D.H. *et al.* (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, **288**, 911–940.

Mignone,F. *et al.* (2002) Untranslated regions of mRNAs. *Genome Biol.*, **3**, REVIEWS0004.

Nussinov,R. *et al.* (1978) Algorithms for loop matchings. *SIAM J. Appl. Math.*, **36**, 68–82.

Park,C. *et al.* (2013) Differential requirements for mRNA folding partially explain why highly expressed proteins evolve slowly. *Proc. Natl. Acad. Sci. U. S. A.*, **110**, E678–E686.

Plotkin,J.B. and Kudla,G. (2011) Synonymous but not the same: the causes and consequences of codon bias. *Nat. Rev. Genet.*, **12**, 32–42.

Raab,D. *et al.* (2010) The GeneOptimizer Algorithm: using a sliding window approach to cope with the vast sequence space in multiparameter DNA sequence optimization. *Syst. Synth. Biol.*, **4**, 2015–2025.

Robbins-Pianka,A. *et al.* (2010) The mRNA landscape at yeast translation initiation sites. *Bioinformatics*, **26**, 2651–2655.

Salis,H.M. *et al.* (2009) Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.*, **27**, 946–950.

Sharp,P.M. and Li,W.H. (1987) The codon adaptation Index–a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res.*, **15**, 1281–1295.

UniProt Consertium (2015) UniProt: a hub for protein information. *Nucleic Acids Res.*, **42**, D204–D12.

Vega Laso,M.R. *et al.* (1993) Inhibition of translational initiation in the yeast *Saccharomyces cerevisiae* as a function of the stability and position of hairpin structures in the mRNA leader. *J. Biol. Chem.*, **268**, 6453–6462.

Wan,Y. *et al.* (2014) Landscape and variation of RNA secondary structure across the human transcriptome. *Nature*, **505**, 706–709.

Zuker,M. and Stiegler,P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, **9**, 133–148.

Zur,H. and Tuller,T. (2012) Strong association between mRNA folding strength and protein abundance in *S. cerevisiae*. *EMBO Rep.*, **13**, 272–277.