Published online 28 September 2017

Nucleic Acids Research, 2017, Vol. 45, No. 22 ***e182***
doi: 10.1093/nar/gkx859

Optimizing DNA assembly based on statistical language modelling

Gang Fang[1,2,*], Shemin Zhang[3] and Yafei Dong[4]

[1]Institute of Advanced Cyberspace Technology, Guangzhou University, Guangzhou 510006, China, [2]Genetic Engineering Laboratory, School of Biological and Environmental Engineering, Xi'an University, Xi'an 710065, China, [3]School of Mathematics and Computer Science, Shaanxi University of Technology, Hanzhong 723001, China and [4]College of life sciences, Shaanxi Normal University, Xi'an 710119, China

Received December 09, 2015; Revised August 31, 2017; Editorial Decision September 05, 2017; Accepted September 16, 2017

ABSTRACT

By successively assembling genetic parts such as BioBrick according to grammatical models, complex genetic constructs composed of dozens of functional blocks can be built. However, usually every category of genetic parts includes a few or many parts. With increasing quantity of genetic parts, the process of assembling more than a few sets of these parts can be expensive, time consuming and error prone. At the last step of assembling it is somewhat difficult to decide which part should be selected. Based on statistical language model, which is a probability distribution P(s) over strings S that attempts to reflect how frequently a string S occurs as a sentence, the most commonly used parts will be selected. Then, a dynamic programming algorithm was designed to figure out the solution of maximum probability. The algorithm optimizes the results of a genetic design based on a grammatical model and finds an optimal solution. In this way, redundant operations can be reduced and the time and cost required for conducting biological experiments can be minimized.

INTRODUCTION

With the development of synthetic biology, it has become necessary to develop tools and methodologies to streamline the design of custom genetic constructs (1). Gene expression studies, gene network studies, protein expression vector design and metabolic engineering are some of applications of this technology (2,3). There are web-based applications such as GenoCAD to fill the needs of these scientific studies. GenoCAD is built upon a solid computational linguistic foundation, and can be used to design synthetic genetic constructs (4). Yet, it captures design strategies of synthetic genetic constructs in the form of grammatical models (5). Promoters, ribosome-binding sites (RBS), genes and terminators are all categories of parts that are needed for designing complex genetic constructs (6–8). Decomposing biological sequences into functional modules as genetic parts is one of the ways to update synthetic biological database (9). There are some assembly standards when assembling these parts into genetic constructs. The BioBrick Foundation (BBF) has been instrumental in promoting the BioBrick standard. A BioBrick compliant part is a DNA fragment flanked by a prefix and a suffix sequence having specific restriction sites (10,11). Two BioBrick parts can be assembled by using a specific series of restriction digestions and ligations independent of the parts sequences. Theoretically, any set of genetic parts compliant with the same standard can be assembled by using specific restriction and ligation enzymes.

When assembling genetic parts into genetic constructs, researchers are often unsure of choosing a part in a particular category. Thus, the molecular biological experiment of assembling is always time-consuming, expensive, and error-prone. In order to reduce the time and cost of assembling, researchers and engineers develop robotic platforms that can help automate the process of assembling many multi-kilobase genetic constructs. The determination of an optimal assembly process can be totally automated by dynamic programming algorithms, without experiences (12). In other robotic platform, a user can design a synthetic construct by successively selecting design rules to transform the structure of the design. Finally, the user can select specific parts to complete the design (13). However, increasing number of genetic parts is being imported. At the last step of assembling, users are always puzzled in choosing a suitable part from few sets of categories (Figure 1). The objective of this study was to overcome this difficulty. Here, we introduce statistical language model (SLM), which can help streamline the assembling process. The first goal of SLM is to build a statistical language model that can estimate the distribution of natural language as accurate as possible (14). The original (and is still the most important) application of SLMs is speech recognition, but SLMs also play a vital role in various other natural language applications as diverse as machine translation, part-of-speech tagging, intelligent in-

*To whom correspondence should be addressed. Tel: +86 18092361338; Email: yuxiangqd@163.com

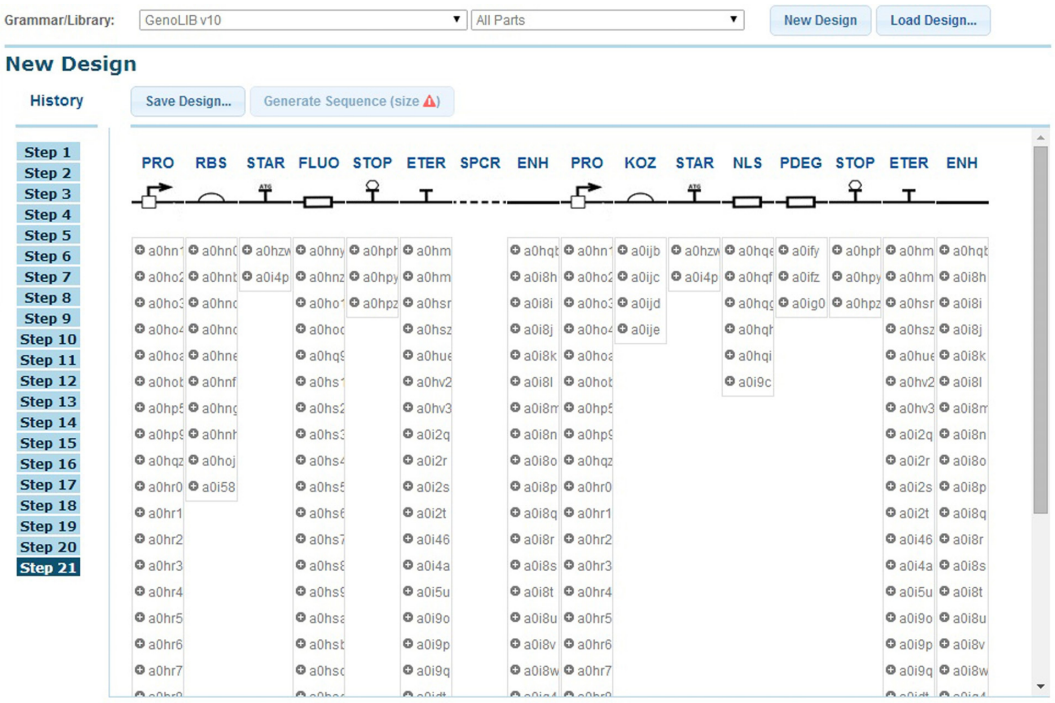

Figure 1. At the last step, it's always difficult to choose a suitable part.

put method and Text To Speech system. The statistical language model in this paper is based on the statistical parameters coming from BioBrick standard parts and widely used plasmids. After transforming the assembling process into the mathematic model, a dynamic programming algorithm can be performed to choose suitable parts composing the final genetic construct. The algorithm takes experience of former iGEM design and some widely used plasmids into account to reduce the cost, time and errors of the assembling process. This method can not only optimize the result of genetic design from other robotic platform but also can help design new projects by considering former experience.

MATERIALS AND METHODS

We use link http://parts.igem.org/das/parts/entry_points/ to download the entry points to the parts that we want to analyze in June 2014. The version of this file published at that time included 7242 parts. A Perl script was developed to parse out the content of each part from the link http://parts.igem.org/das/parts/features/?segment=part#. We then decomposed them into structured data format, which could be imported into a MySQL database. After being imported into a MySQL database, 75744 features were parsed out from these parts. The parts include both basic parts (e.g. promoter and RBS) and composed parts, which include multiple basic parts (e.g. device, project and composite). The basic parts include categories of Regulatory, RBS, Coding, Terminator and Plasmid Backbone. We queried the MySQL database to extract the basic parts and counted their usage in composed parts. We also developed Perl script and SQL sentences to analyze composed parts and counted the usage of two (parts pair), three (parts triple), and four (parts quadruple) adjacent basic parts in them. By querying the MySQL database, we extracted a set of 1682 basic parts compliant with RFC 23 standard (15). It means that the sequence of these basic parts does not include any of the restriction sites used by the assembly standard. These 1682 basic parts include 405 promoters, 42 RBSs, 57 terminators and 1178 genes. We used these basic parts to design some genetic constructs. The usage frequencies of basic parts, parts pair, parts triple, and parts quadruple in the dataset were calculated.

We also downloaded genetic parts from GenoLIB database and developed Perl scripts and SQL sentences to analyze them. 1633 parts which consisted of 84 categories were included and their usages in 1750 commonly used plasmids were counted. The usages of parts pair, parts triple and parts quadruple were also counted for optimizing parts assembly based on SLM. Readers can be referred to homepage of GenoCAD (www.genocad.com) for more information of these parts and categories.

Mathematic model

For example, at the last step of GenoCAD design, every icon has its option. It is somewhat difficult for the designer to choose the most suitable part to complete the design (Figure 1). There are too many combinations of parts to form different genetic constructs. It is impossible to exam every combination with wet experiment. And adopting a reasonable mathematical model to depict the genetic construct will facilitate selecting suitable parts (16). To overcome this problem, statistical language model (SLM) is introduced. In this model, whether a sentence (S) is meaningful and reasonable is based on the probability it will occur. A sentence

(S) is composed of a sequence of words. Here S is a genetic construct that is made of basic parts, and the words are these basic parts. Now, $S = part_1, part_2, \ldots, part_n$ and we need to know its $P(S)$—the probability that it will occur. Thus,

$$P(S) = P(part_1, part_2, ..., part_n) \quad (1)$$

According to conditional probability formula

$$\begin{aligned} &P(part_1, part_2, ...part_n) \\ &= P(part_1) \cdot P(part_2 | part_1) \cdot P(part_3 | part_1, part_2) \quad (2) \\ &\cdots P(part_n | part_1, part_2, ..., part_{n-1}) \end{aligned}$$

In formula (2), $P(part_1)$ means that the probability $part_1$ appears in the design. $P(part_2|part_1)$ denotes the probability that $part_2$ appears with $part_1$ prior to it. According to formula (2), the probability $part_n$ appears is determined by all the parts that appear prior to it. The $P(part_1)$ and $P(part_2|part_1)$ are easy to calculate, but calculating $P(part_3|part_1,part_2)$ is difficult. Further, calculating $P(part_n|part_1,part_2,\ldots,part_{n-1})$ is very difficult, because much more variables are involved in it. The conditions are too complex to gauge. We believe, based on Markov Hypothesis, that the probability that a part will occur is only concerned with the part prior to it. Thus, formula (2) can be simplified as:

$$\begin{aligned} &P(S) \\ &= P(part_1) \cdot P(part_2 | part_1) \cdot P(part_3 | part_2) \quad (3) \\ &\cdots P(part_i | part_{i-1}) \cdots P(part_n | part_{n-1}) \end{aligned}$$

This is the Bigram of statistical language model. However, when assembling genetic parts, whether a gene can be expressed effectively is not only concerned with its RBS but also with its promoter and plasmid backbone. Therefore, we believe that the probability that a part will occur in an S is concerned with the adjacent two or three parts prior to it. Thus, formula (2) can be simplified as:

$$\begin{aligned} &P(S) \\ &= P(part_1) \cdot P(part_2 | part_1) \cdot P(part_3 | part_1, part_2) \quad (4) \\ &\cdots P(part_i | part_{i-2}, part_{i-1}) \cdots P(part_n | part_{n-2}, part_{n-1}) \end{aligned}$$

Formula (4) is the 3-gram model of statistical language model. Based on this theory, the 4-gram model can be established. According to conditional probability formula:

$$P(part_i | part_{i-2}, part_{i-1}) = \frac{P(part_{i-2}, part_{i-1}, part_i)}{P(part_{i-2}, part_{i-1})} \quad (5)$$

We use usage frequencies of parts pair, parts triple, and usage frequencies of basic parts to estimate $P(part_{i-2}, part_{i-1}, part_i)$, $P(part_{i-2}, part_{i-1})$ and $P(part_i)$, respectively.

$$P(part_{i-2}, part_{i-1}, part_i) \approx \frac{Count(part_{i-2}, part_{i-1}, part_i)}{Count(all\_parts)}$$

$$P(part_{i-2}, part_{i-1}) \approx \frac{Count(part_{i-2}, part_{i-1})}{Count(all\_parts)}$$

$$P(part_i) \approx \frac{Count(part_i)}{Count(all\_parts)}$$

According to formula (5) and the above formulas,

$$P(part_i | part_{i-2}, part_{i-1}) \approx \frac{Count(part_{i-2}, part_{i-1}, part_i)}{Count(part_{i-2}, part_{i-1})} \quad (6)$$

$$P(part_i | part_{i-1}) \approx \frac{Count(part_{i-1}, part_i)}{Count(part_{i-1})}$$

In this way, any component in formula (4) can be calculated.

At the last step of design (Figure 1), there are too many combinations of basic parts to complete the design. Which one is the most reasonable and meaningful? We believe that the part combination with the largest probability of occurrence is the answer. We have all the candidate paths, and a path will result in an S ($a\ path = a\ S = part_1, part_2, \ldots, part_n$). The best path is represented by $PATH$.

$$PATH = \arg\max_{all\_S}(P(S))$$

To avoid memory overflow when performing the algorithm on a computer, we considered the *log* of $P(S)$.

$$\begin{aligned} &PATH \\ &= \arg\max_{all\_S}(\log P(S)) \\ &= \arg\max_{all\_S}(\log(P(part_1) \times P(part_2 | part_1) \times \prod_{i=3}^{n} P(part_i | part_{i-2}, part_{i-1}))) \quad (7) \\ &= \arg\max_{all\_S}(\log P(part_1) + \log P(part_2 | part_1) + \sum_{i=3}^{n} \log P(part_i | part_{i-2}, part_{i-1})) \end{aligned}$$

Based on formula (6), we derived the following formulas

$$P(part_i | part_{i-2}, part_{i-1}) = \frac{Count(part_{i-2}, part_{i-1}, part_i)}{Count(part_{i-2}, part_{i-1})}$$

$$P(part_2 | part_1) = \frac{Count(part_1, part_2)}{Count(part_1)}$$

$$P(part_1) = \frac{Count(part_1)}{Count(all\_parts)}$$

Since we extracted the dataset from a relatively sparse corpus, the zero-frequency problem would arise when parts pair, parts triple never occurred in the training corpus. The zero-frequency problem means when some parts pairs and triples never appear in a corpus, their *Count* will be zero. Then, according to above formulas the probability they appear is zero. This circumstance will cause troubles in calculating $P(S)$ and selecting the $PATH$. To overcome this, we use Add-one (Laplace) Smoothing (17). Thus, these formulas should be represented as:

$$P(part_2 | part_1) = \frac{Count(part_1, part_2) + 1}{Count(part_1) + \mathrm{N}} \quad (8)$$

$$P(part_i | part_{i-2}, part_{i-1}) = \frac{Count(part_{i-2}, part_{i-1}, part_i) + 1}{Count(part_{i-2}, part_{i-1}) + W} \quad (9)$$

In formulas (8) and (9), N is the number of bi-gram (parts pair) and W is the number of 3-gram (parts triple). Formulas (8) and (9) were used to fill the corresponding component in formula (7). Based on same theory, a 4-gram model of statistical language model can be established and corresponding probability can be calculated. The resulted $PATH$ was the S with the largest probability of occurrence in all candidate paths. We used dynamic programming algorithm to figure out the $PATH$ from all candidates.

Table 1. The grammatical model used in this study

Rule	Comments	Parent term	Child term
1	Transform a cassette (Cass) into two cassettes (Cass)	Cass	Cass-Cass
2	Reverse the sequence orientation of a cassette (Cass)	Cass	[Cass]
3	Transform a cassette (Cass) into a promoter (PRO), a cistron (Cis), and a terminator (TERM)	Cass	PRO-Cis-TERM
4	Transform a cassette (Cass) into a promoter (PRO), a cistron(Cis)	Cass	PRO-Cis
5	Transform a cistron (Cis) into two cistrons (Cis)	Cis	Cis-Cis
6	Transform a cistron (Cis) into a rbs (RBS) and a gene (GEN)	Cis	RBS-GEN
7	Transform a terminator (TERM) into two terminators (TERM)	TERM	TERM-TERM
8	Transform a gene (GEN) into two genes (GEN)	GEN	GEN-GEN

Algorithm

Next, we need to find a path in the lattice in Figure 1. This path is composed of a series of parts and will be the S with the largest probability. It reflects how formula (7) can be solved. The algorithm of 3-gram model is presented as an example, it originates from the Viterbi algorithm (18) and consists of three steps:

Firstly, a candidate lattice was built (Figure 2). Every icon (category) corresponds to one column, and every node in a column corresponds to a basic part. At the start and end of the lattice, BEG and END columns were added. In these two columns, two virtual nodes of B and E, respectively, were added (Figure 2). Every node is a triple-tuple $< name, V, P >$, and the first element *name* was filled with basic part name.

Secondly, the lattice was filled (Figure 3). In the lattice from left to right, for every node of a triple-tuple $< name, V, P >$, the V and P are calculated and filled. V was filled with the maximum value selected from combining operation of three nodes in three adjacent columns. P would then store the address of the node prior to it, where V is derived from via combining operation.

(1) First column, for the B node, let $V = 0$ and $P =$ NULL.
(2) Second column, every node $< name, V, P >$ ($name \in$ {I0500, R0011, …, R0040, …}) was combined with B node, and its V and P were calculated as:

$$V = V_B + \log P(part) = \log P(part), \cdot P = address\_of\_B$$

(3) Third column, every node $< name, V, P >$ ($name \in$ {R0032, R0034, …, R0041, …}) was combined with every node in the second column, and its V and P were calculated as:

$$V = \max\{\log P(part \,|\, part_{prior})\},$$
$$P = address\_where\_V\_comes\_from$$

(4) Fourth column, every node $< name, V, P >$ was combined with every node in the previous two columns, and its V and P were calculated as:

$$V_{Current} = \max\{V_{Before} + V_{Previous} + \log P(part \,|\, part_{Before}, part_{Previous})\}$$

$$P_{Current} = Address\_of\_V_{Previous}\_belong$$

$$P_{Previous} = Address\_of\_V_{Before}\_belong$$

(5) Step 4 was repeated; every node in the current column was combined with every node in the previous two columns, and its V and P were calculated.
(6) In the END column, V is the maximum value selected from the nodes in the previous column, P would store the address of the node where V is derived form.

Thirdly, the *PATH* was recalled and obtained (Figure 4). Starting from node E, the P prior to it was continually searched (Figure 4). Finally, the *PATH* with the largest probability was found, and the resulted S was the genetic construct that was designed. In same way, a dynamic programming algorithm can be implemented in a 4-gram model and the corresponding *PATH* can be figured out. If the length of S is L, and the maximum node number in a column is D, the algorithm complexity of this algorithm in a 3-gram model will be $O(L \cdot D^3)$ and $O(L \cdot D^4)$ in a 4-gram model, and the algorithm complexity of exhaustive algorithm is $O(D^L)$.

RESULTS

The general methodology of developing grammars to model the structure of synthetic genetic constructs has been previously described in details (4,19). The basic grammar is clear: **PRO** (promoter)-**RBS** (ribosome-binding sites)-**GEN** (genes)-**TERM** (terminator). The grammatical model used in this study is similar to the context-free grammar (CFG) (19), but has new rewriting rules to allow protein fusion. The full grammatical model is described in Table 1. We also used GenoLIB grammar to design new genetic constructs, the database and the grammar also have been described before in details (9).

To demonstrate how to assemble BioBrick parts to form a functional biosynthetic system, we selected the banana odor biosynthetic system (http://parts.igem.org/Part:BBa_J45900), designed and implemented by the MIT iGEM team in 2006. The system contains two expression cassettes: one with *BAT2* and *THI3* genes, which produces isoamyl alcohol; and the other catalyzes the conversion of the cellular metabolite leucine to isoamyl acetate or banana odor. We can design the system according to the grammatical model.

Firstly, we needed a **Cass**, and then applied Rule 1. The design became **Cass-Cass**, and then we applied Rule 4 to the first **Cass** and Rule 3 to the second **Cass**. The design became **PRO-Cis-PRO-Cis-TERM**. We applied Rule 5 to the first **Cis** and Rule 6 to the second **Cis**. The design became **PRO-Cis-Cis-PRO-RBS-GEN-TERM**. Next, we apply Rule 6 to **Cis**, and the design becomes **PRO-RBS-GEN-RBS-GEN-PRO-RBS-GEN-TERM**. Finally, we applied Rule 7, after which the final design was **PRO-RBS-GEN-RBS-GEN-PRO-RBS-GEN-TERM-TERM**. Before we input the de-

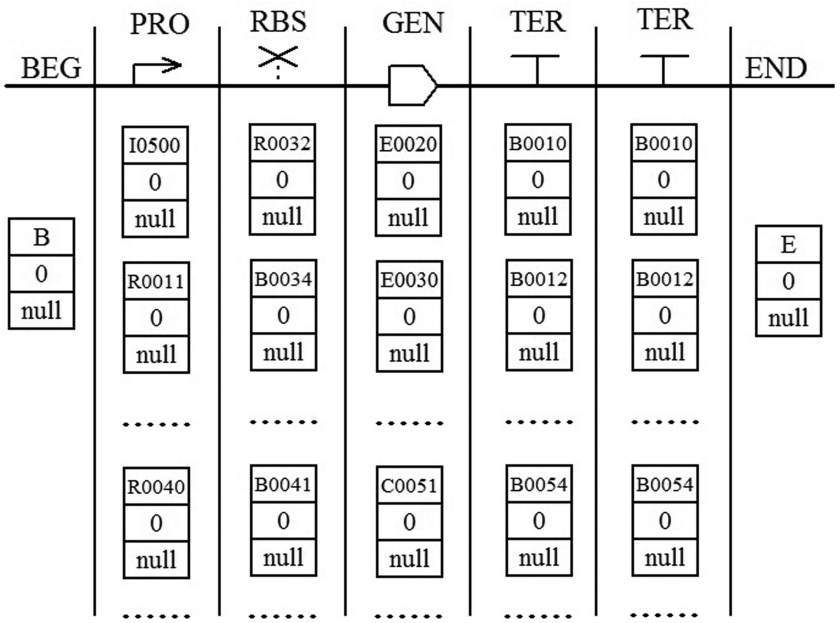

Figure 2. Building a candidate lattice.

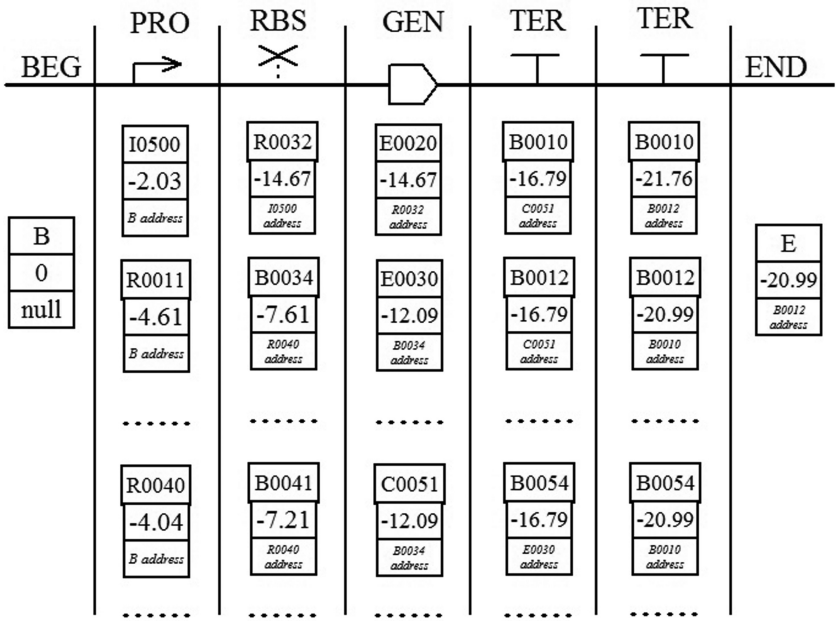

Figure 3. Filling the lattice.

sign, the genes we want to express should be determined. Then, the input design was

PRO RBS J45008 **RBS** J45009 **PRO RBS** J45014 **TERM TERM**

The assembling algorithm was implemented by a Perl script, and then the 3-gram model algorithm recommended the parts series:

R0040-B0030-J45008-B0030-J45009-R0040-B0030-J45014-B0010-B0012.

The 4-gram model algorithm recommended:

R0011-B0030-J45008-B0030-J45009-R0040-B0030-J45014-B0010-B0012.

This is the actual part that banana odor biosynthetic system consists of.

For another example, we selected the wintergreen odor biosynthetic system (http://parts.igem.org/Part:BBa_J45700), also designed and implemented by the MIT iGEM team in 2006. The system consists of two expression cassettes: one produces salicylate acid from the cellular

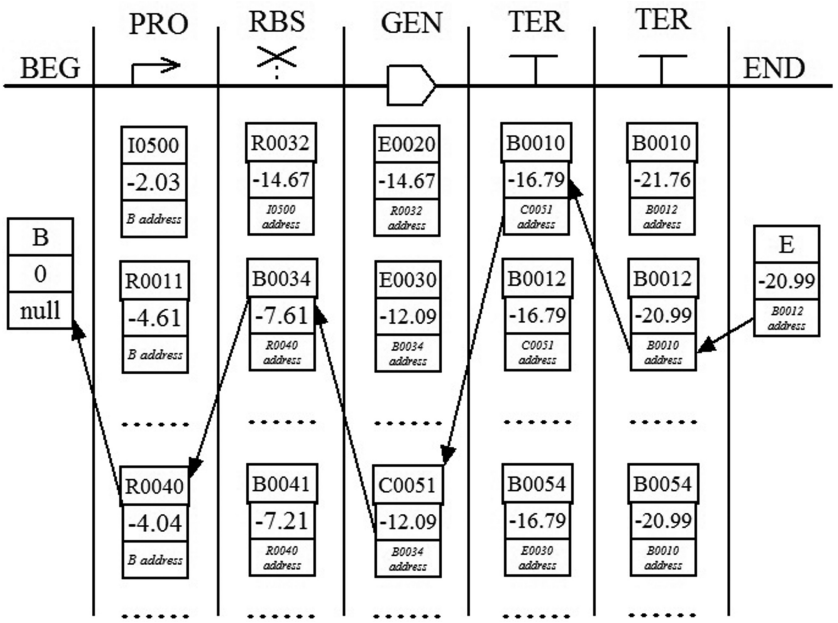

Figure 4. Recalling and obtaining the *PATH*.

metabolite; and the other converts the salicylic acid to methyl salicylate, which produces the wintergreen odor. According to the grammatical model, the design started with a **Cass**. Then, on applying Rule 1, the design became **Cass-Cass**. We applied Rule 3 to both **Cass**, and the design became **PRO-Cis-TERM-PRO-Cis-TERM**. We applied Rule 6 to both **Cis**, and the design became **PRO-RBS-GEN-TERM-PRO-RBS-GEN-TERM**. Finally, we applied Rule 7, and the final design was **PRO-RBS-GEN-TERM-TERM-PRO-RBS-GEN-TERM-TERM**. The genes to be expressed should be determined before input. The input design was

PRO RBS J45004 **TERM TERM PRO RBS** J45017 **TERM TERM**

The assembling algorithm of 3-gram model recommended:

R0040-B0030-J45004-B0010-B0012-R0062-B0030-J45017-B0010-B0012.

The 4-gram model recommended:

R0040-B0030-J45004-B0010-B0012-R0011-B0030-J45017-B0010-B0012.

Two examples repeated the combination R0040-B0030. According to expertise in this field, R0040-B0030 together with Biobrick scars at each junction would result in a 93bp direct repeat, which might be expected to make the system unstable. To avoid this bias, we modified the parameters by deleting combinations with less usage frequency (less than four times in parts usage, less than three times in parts pair usage, less than three times in parts triple usage). This is a general procedure to correct some deviations in assembling results. Then, in the first example the assembling algorithm of 3-gram model recommended:

R0040-B0030-J45008-B0030-J45009-I0500-B0032-J45014-B0010-B0012.

The 4-gram model algorithm also recommended:

R0011-B0030-J45008-B0030-J45009-R0040-B0030-J45014-B0010-B0012.

In the second example the 3-gram model recommended:

R0040-B0032-J45004-B0010-B0012-R0062-B0032-J45017-B0010-B0012.

The 4-gram model recommended:

R0040-B0032-J45004-B0010-B0012-R0011-B0032-J45017-B0010-B0012.

And this is the actual part that wintergreen odor biosynthetic system consists of.

The third example, we used the **sascsc** design in Public Designs of GenoCAD (Figure 5). This design used GenoLIB v10 grammar and included All Parts. When inputting the design **PRO RBS CDS ETER SPCR ENH VREP ENH**, the assembling algorithm of 3-gram model recommended:

T7 promoter. ribosome binding site-003. DHFR. MAS terminator. SPCR. hr5 enhancer. ORF1629-004. hr5 enhancer.

The 4-gram model recommended:

AmpR promoter-009. ribosome binding site-003. DHFR. MAS terminator. SPCR. hr5 enhancer. ORF1629-004. hr5 enhancer.

When we optimized it with the Most Popular Parts Library in GenoLIB database. Both 3 and 4-gram model recommended the result:

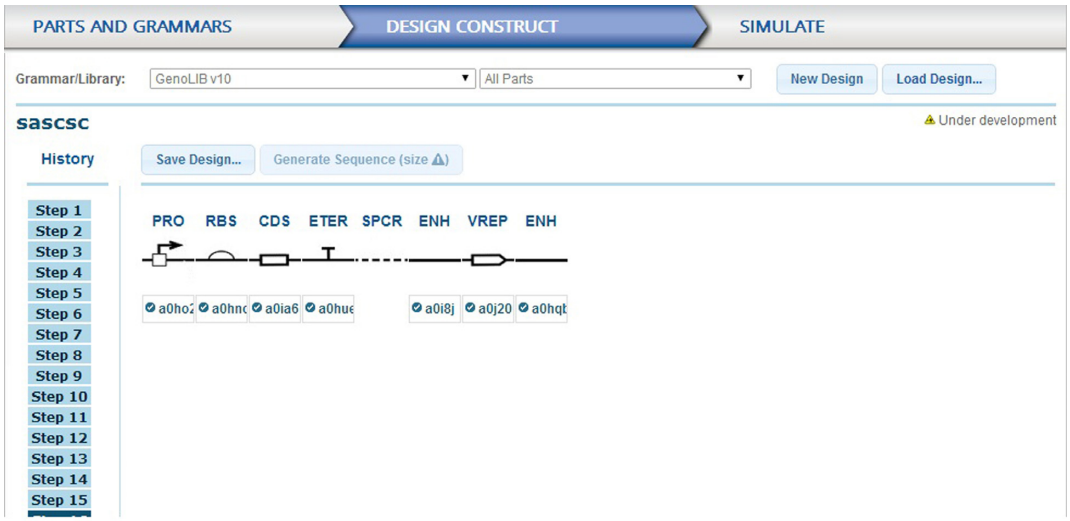

Figure 5. The **sascsc** design in GenoCAD.

CMV promoter-009. ribosome binding site-002. DHFR. rabbit globin terminator-001. SPCR. CMV enhancer-003(Recommended). ORF1629-001. CMV enhancer-003(Recommended).

The fourth example (Figure 6), we developed a new design with GenoLIB v10 grammar and optimized it with the Most Popular Parts Library. Both 3- and 4-gram model recommended the result:

CMV promoter-009. kozak sequence-003. ATG. S-Tag-002. stop-003. rrnB T1 terminator-002. SPCR. RSV promoter-003. ribosome binding site-002. ATG. gag(truncated)-001. stop-003. rabbit globin terminator -001.

When we optimized the design with the All Parts Library in GenoLIB database, the 3-gram model recommended:

tet promoter-002. kozak sequence-003. ATG. GST-002. stop-003. T3Te terminator-001. SPCR. cat promoter-004(Recommended). RBS. ATG. TrxA-003. stop-003. MAS terminator.

And the 4-gram model recommended:

AmpR promoter-009. kozak sequence-002. ATG. GST-002. stop-003. T3Te terminator-001. SPCR. p10 promoter-001. ribosome binding site-003. ATG. M13 gene II. stop-003. MAS terminator.

In the first example, the 4-gram model recommended the same assembly to the original. The 3-gram model recommended assembly different from the original. In the second example, the 4-gram model recommended an assembly similar to the original composition. The 3-gram model recommended assembly quite different from the original. After modifying the parameters, in both examples the 3-gram model recommended an assembly similar to the original and the 4-gram model recommended the same one to the original. Experienced users can modify the parameters to optimize assembling results. In the first example that simulating iGEM design in **Supplementary Data**, both 3 and 4-gram model recommended assemblies different from the originals. After modifying the parameters, the 4-gram model recommended a same one to the original. When we develop more complicated designs (the other three examples simulating iGEM design), the algorithm recommended quite different results (**Supplementary Data**). One reason is the corpus is fairly sparse, more successful assemblies are needed to enrich the corpus. And the other important reason is that when the designs become complicated a more advanced grammar will be needed. We use GenoLIB v10 grammar in GenoCAD to the third and fourth examples, and the third one is also a public design. The algorithm recommended some assembly results for consideration. If we need some other options, we can exclude some parts and repeat the algorithm. It will recommend some other optimized assemblies for consideration. If we have known that some parts are definitely connected, we can determine them first (or evaluate them higher) then implement the algorithm. Readers can be referred to **Supplementary Data** for these applications and more examples. Theoretically, the 4-gram model will recommend a more reasonable assembly result than 3 and Bi-gram model. But we think in some cases it must be tested by a wet experiment. The algorithm can be iterated to yield different optimized results for consideration. If we develop new project according to a grammar and carry out the algorithm at the last step, the algorithm will yield an optimized assembly based on experience. Students enrolled in the iGEM competition may be an important group of potential user of this method, and the BioBrick grammar has been developed with this group in mind.

DISCUSSION

This paper presents a statistical language model for synthetic biological parts assembling. After converting syn-

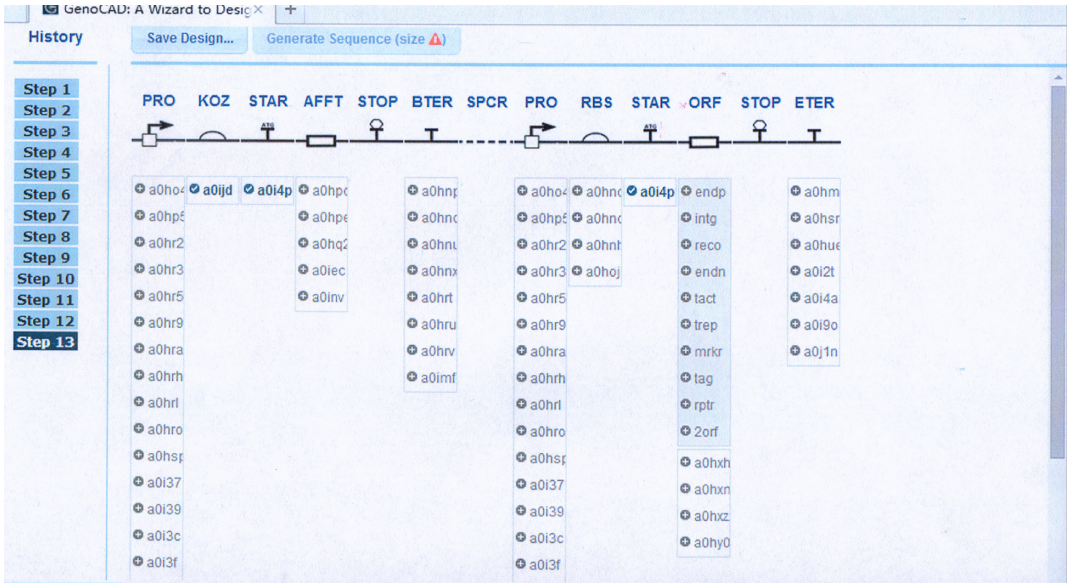

Figure 6. A GenoCAD design to show the optimized results.

thetic biological parts assembly process into a 3- or 4-gram model, a dynamic programming algorithm was carried out to select an optimized result. The method can not only be applied to optimize the assembly of a design in a synthetic biological robotic platform such as GenoCAD, but can also be independently applied to automate the DNA assembly process in synthetic biology. After entering categories of synthetic biological parts according to a grammar, the algorithm automatically selects suitable parts to form a reasonable construct by considering successful experiences. In this way, redundant operations can be reduced, and the time and cost required for conducting biological experiments can be minimized. As described previously, this method is based on 3- or 4-gram model. It indicates every part involved in the assembly process is concerned with the adjacent two or three parts prior to it. But in real world DNA assembly process, for an example, whether a gene can be expressed effectively is not only related to its promoter, RBS but also its plasmids backbone and other regulating sequences. In order to simulate real world assembly process, models higher than 4-gram should be introduced. These models indicate every part involved in the assembly process is concerned with more than three parts prior to it. But in these higher-gram models the conditional probability is very difficult to calculate. When $N = 5$ or more, though the accuracy in other natural language applications such as machine translation, part-of-speech tagging, intelligent input method will increase significantly, powerful computer will be needed (14).

When calculating the conditional probability, we used Add-one (Laplace) smoothing to overcome zero-frequency problem. However, it is always not a good choice and the disadvantages of Add-one (Laplace) smoothing are well-known: it allows considerable amount of the probability space to unseen events, and is poor at predicting the actual probabilities of bi-grams (17). It was used just for simplicity, considering that any two parts compliant with the same standard can be connected. We intend to develop 5- or more-gram models, and expand the corpus to simulate the assembly process more reasonably. Other smoothing technologies such as Good-Turing Smoothing, Katz backoff, Interpolation Smoothing (20,21) will be considered to improve the mathematical model. Some parts are overwhelmingly likely to be returned in any analysis. One of the reasons is that 'noises' in the corpus cause results deviating, and we can delete these 'noises' to correct the deviation. We employed commonly used combinations with higher usage frequency (more than three times in parts usage, more than two times in parts pair and triple usage, and all the parts quadruple) in the iGEM dataset, and we defined these combinations were 'successful' or 'good' words and the others was useless 'noises'. We downloaded a relatively sparse corpus from the iGEM website, and derived a fairly sparse corpus from widely used plasmids. We also counted the usage of parts and two or more continuous adjacent parts. When developing new projects, some parts pairs, parts triples and parts quadruples never occur in the sparse corpus. These circumstances always raise the zero-frequency problem, and cause optimized results deviating. Besides improving smoothing technology, expanding the corpus is absolutely necessary. But expanding corpus needs describing the nature of features and parts in a unified way. In other words, same sequence should have same name and same description. This is still a more difficult issue. This can be solved by the development of an ontology giving the community a common controlled vocabulary to describe the features. And developing the Synthetic Biology Open Language will promote this process.

Based on previously described linguistic models of synthetic DNA sequences, the paper presents a computational supplement for the AutoCAD platform of Synthetic Biology. It also has raised an important question of a too wide choice being offered at the final stage of the design. It is fairly important to take into account success of the previ-

ous assemblies when creating a new design. For an inexperienced user or a new person to Synthetic Biology, it will be difficult to make a right choice of parts to be used in the assembly. The algorithm help users by suggesting the most common combinations of parts based on a body of previously analyzed and successful constructs. Besides optimizing BioBrick parts assembling, by using GenoLIB grammar and dataset in GenoCAD to create designs and optimize the assembly it demonstrates that the method can be used more practically and universally. The newly developed feature will facilitate popularizing of Synthetic Biology to a wider community and might help to erase inconsistency and jargon used in this field. On the other hand, when collecting data from previous assemblies we know usage frequencies of parts and parts connections. This will enrich our knowledge of synthetic biological assemblies and favor the data-driven feature of biotechnology. Next step, more successful assemblies will be taken into account and the algorithm will be improved to enable it useful for a wider scientific community.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

Authors acknowledge the BBF and the iGEM community for contributing the RFCs and BioBrick parts upon which this project was developed. Thanks for Prof. Jean Peccoud from Colorado State University and Mandy Wilson from Virginia Bioinformatics Institute guiding authors collecting the data. We would like to thank anonymous reviewers for their valuable comments to improve this paper.

FUNDING

Natural Science Foundation of China [61173113, 61572302]. Funding for open access charge: Natural Science Foundation of China [61173113].
Conflict of interest statement. None declared.

REFERENCES

1. Goler,J.A., Bramlett,B.W. and Peccoud,J. (2008) Genetic design: rising above the sequence. *Trends Biotechnol.*, **26**, 538–544.
2. Graslund,S., Nordlund,P., Weigelt,J., Hallberg,B.M., Bray,J., Gileadi,O., Knapp,S., Oppermann,U., Arrowsmith,C., Hui,R. *et al.* (2008) Protein production and purification. *Nat. Methods*, **5**, 135–146.
3. Ghaemmaghami,S., Huh,W.K., Bower,K., Howson,R.W., Belle,A., Dephoure,N., O'Shea,E.K. and Weissman,J.S. (2003) Global analysis of protein expression in yeast. *Nature*, **425**, 737–741.
4. Czar,M.J., Cai,Y. and Peccoud,J. (2009) Writing DNA with GenoCAD. *Nucleic Acids Res.*, **37**, W40–W47.
5. Cai,Y., Wilson,M.L. and Peccoud,J. (2010) GenoCAD for iGEM: a grammatical approach to the design of standard-compliant constructs. *Nucleic Acids Res.*, **38**, 2637–2644.
6. Isaacs,F.J., Dwyer,D.J., Ding,C., Pervouchine,D.D., Cantor,C.R. and Collins,J.J. (2004) Engineered riboregulators enable posttranscriptional control of gene expression. *Nat. Biotechnol.*, **22**, 841–847.
7. Gardner,T.S., Cantor,C.R. and Collins,J.J. (2000) Construction of a genetic toggle switch in Escherichia coli. *Nature*, **403**, 339–342.
8. Atkinson,M.R., Savageau,M.A., Myers,J.T. and Ninfa,A.J. (2003) Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in Escherichia coli. *Cell*, **113**, 597–607.
9. Adames,N.R., Wilson,M.L., Fang,G., Lux,M.W., Glick,B.S. and Peccoud,J. (2015) GenoLIB: a database of biological parts derived from a library of common plasmid features. *Nucleic Acids Res.*, **43**, 4823–4832
10. Arkin,A. (2008) Setting the standard in synthetic biology. *Nat. Biotechnol.*, **26**, 771–774.
11. Canton,B., Labno,A. and Endy,D. (2008) Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.*, **26**, 787–793.
12. Densmore,D., Hsiau,T.H.C., Batten,C., Kittleson,J.T., DeLoache,W. and Anderson,J.C. (2010) Algorithms for automated DNA assembly. *Nucleic Acids Res.*, **38**, 2607–2616.
13. Coll,A., Wilson,M.L., Gruden,K. and Peccoud,J. (2015) Rule-based design of plant expression vectors using GenoCAD. *PLoS ONE*, **10**, e0132502.
14. Jelinek,F. (1998) *Statistical Methods for Speech Recognition (Language, Speech, and Communication)*, MIT Press.
15. Phillips,I.E. and Sliver,P.A. (2006) A New Biobrick Assembly Strategy Designed for Facile Protein Engineering. DSpace@MIT, https://dspace.mit.edu/handle/1721.1/32535.
16. Ellis,T., Wang,X. and Collins,J.J. (2009) Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat. Biotechnol.*, **27**, 465–471.
17. Chen,S.F. and Goodman,G. (1999) An empirical study of smoothing techniques for language ing. *Comp. Speech Lang.*, **13**, 359–394.
18. Viterbi,A.J. (2006) A personal history of the Viterbi algorithm. *IEEE Signal Process. Mag.*, **23**, 120–142.
19. Cai,Y., Hartnett,B., Gustafsson,C. and Peccoud,J. (2007) A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*, **23**, 2760–2767.
20. Huang,F.L., Yu,M.S. and Hwang,C.Y. (2013) An empirical study of good-turing smoothing for language s on different size corpora of Chinese. *J. Comput. Commun.*, **1**, 14–19.
21. Katz,S.M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Trans. Acoust. Speech Signal Process.*, **35**, 400–401.