doi:10.3969/j.issn.1672-5565.2016.03.08

动态规划算法对 GenoCAD 设计结果的优化

方　刚[1,2]
(1.西安文理学院生物与环境工程学院,西安 710065;
2.西安财经学院信息学院,西安 710100)

摘　要:GenoCAD(www.genocad.com)是一种基于 Web 的免费合成生物学设计软件,用它可以进行表达载体及人工基因网络设计。持续点击代表各种合成生物学标准“零件”的图标,以一种语法进行设计,最后就可以得到由数十个功能片段组成的复杂质粒载体。但是在 GenoCAD 中,每一类的合成生物学标准“零件”数量众多。随着这些标准“零件”的不断开发,其数量也在进一步增加,目前选择合适的“零件”组装成功能性的质粒载体费时费力并且容易发生错误。在进行载体设计的最后阶段,从众多的“零件”中选择合适的往往比较困难。为解决这一问题,本文采用了自然语言处理的统计语言模型,它最初用于自然语言识别,用来估算一组词串成为一个正确语句的概率的大小。本文最后以该模型为基础应用动态规划算法优化质粒载体设计,从众多的选项中找出最优者。利用这一方法可以减少进行生物学实验的冗余操作,从而减少载体构建过程中的花费。

关键词:合成生物学;统计语言模型;动态规划算法;生物学“零件”;GenoCAD
中图分类号:Q291　　**文献标志码**:A　　**文章编号**:1672-5565(2016)03-173-08

Dynamic programming optimization to GenoCAD design

FANG Gang[1,2]
(1.*School of Biological and Environmental Engineering*, *Xi' an University*, *Xi' an* 710065 ,*China*;
2.*School of Information*, *Xi' an University of Finance and Economics*, *Xi' an* 710100,*China*)

Abstract: GenoCAD (www.genocad.com) is a free web-based application that guides the user to design protein expression vector, artificial gene networks and other genetic constructs composed of genetic parts. By successively click icons representing actual genetic parts according to grammatical models, complex genetic constructs composed of dozens of functional blocks can be designed. But at the last step of design, usually every icon representing genetic parts has its option. With the increasing of genetic parts database, more and more parts were imported into GenoCAD library. The process of assembling more than a few of sets of genetic parts can be costly, time consuming and error prone. At the last step of design it is somewhat difficult to make decision 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 solve the problem. The algorithm optimizes the results of GenoCAD design and finds an optimal solution. In this way, redundant operations can be reduced and the time and cost required for conducting biological experiment can be minimized.

Keywords: Synthetic biology; Statistical language model; Dynamic programming; BioBrick; GenoCAD

1　Introduction

With the development of synthetic biology, it has become necessary to develop tools andmethodologies to streamline the design of custom genetic constructs[1]. Gene expression studies, gene network studies, protein expression vector design and metabolic engineering are

收稿日期:2016-04-01;**修回日期**:2016-05-12.
基金项目:国家自然科学基金项目(No.61173113)。
作者简介:方刚,男,副教授,研究方向:生物信息学、合成生物学;E-mail: yuxiangqd@163.com.

some of applications of this technology [2-3]. GenoCAD is a web-based application to fill the needs of these scientific studies. It is built upon a solid computational linguistic foundation and can be used to design synthetic genetic constructs[4]. Yet, its point and-click graphical user interface enables users to design complex constructs in a matter of minutes. GenoCAD captures design strategies of synthetic genetic constructs in the form of grammatical models [5]. It provides a central parts database with each grammar, and the latest GenoLIB grammar comes with a library of 1943 basic genetic parts that come from 2 000 widely used plasmids [6]. As proof, the library was converted into GenoCAD grammar files to allow users to import and customize the library based on the needs of their research projects. Users, who elect to create a GenoCAD personal account, can log in the system to create project-specific parts libraries, upload new parts into their workspace and save designs for later use [5]. Thinking of genetic systems as composed of parts, each with its own function and characteristics, is the design philosophy of GenoCAD. Promoters, ribosome-binding sites (RBS), genes and terminators are all categories of parts that are needed for designing complex genetic constructs[7-9] in GenoCAD. Decomposing biological sequences into functional modules as genetic parts is one of the ways to update the GenoCAD library[6].

When researchers assembling more than a few specific genetic parts from different categories, the process is always costly, time consuming and error prone. In order to streamline this process, some assemblystandards are introduced. 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. 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 thinking of experience[12]. In GenoCAD design, a user can design a synthetic construct by successively selecting design rules to transform the structure of the design. At last select specific parts to complete the design[13]. But more and more genetic parts are imported into GenoCAD. Now, users are always puzzled to choose a suitable part from few sets of categories in the last step of a design. To overcome this, statistical language model (SLM) is introduced in this paper which can help streamline the process of design. 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 input method and Text To Speech system. The statistical language model in this paper is based on the statistical parameters coming from BioBrick standard parts. After transforming the design process into this mathematic model, a dynamic programming algorithm can be carried out to choose suitable parts composing the final genetic construct. The algorithm takes experience of former iGEM design into account to reduce the cost, time and errors of the assembling process. This method can not only optimize the result of GenoCAD design but also can help design new projects by considering former experience.

2 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 7 242 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#. And decomposed them into structured data format, which could be imported into a MySQL database. After imported into a MySQL database, 75 744 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 adjacent basic parts (parts pair) in them. By querying the MySQL database, we extracted a set of 1 682 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 1 682 basic parts include 405 promoters, 42 RBSs, 57 terminators and 1 178 genes. We imported these basic parts into GenoCAD to design new genetic constructs. And the usage frequencies of basic parts and the usage frequencies of parts pair in the dataset can be calculated.

3　Grammar design in GenoCAD

The general methodology of developing grammarsin GenoCAD to model the structure of synthetic genetic constructs has been described detailedly before[4, 16]. The grammar used in this article is similar to the context-free grammar (CFG)[16], but has new rewriting rules to allow protein fusion. We used biobrick_icon_set to represent the categories of basic parts. The full grammar is described in Table 1.

Table 1　The grammar used in this paper

Rule	Comments	Left term	Right term
1	Transform aStart (RFC 23) into a plasmid backbone (Pb), a prefix (Pr), a cassette (Ca) and a suffix (Su)	S	Pb Pr Ca Su
2	Transform a cassette (Ca) into two cassettes (Ca) with a scar (Sc) in between	Ca	Ca Sc Ca
3	Reverse the sequence orientation of a cassette (Ca)	Ca	[Ca]
4	Transform a cassette (Ca) into a promoter (P), a scar (Sc), a cistron (C), a scar (Sc) and a terminator (T)	Ca	P Sc C Sc T
5	Transform a cassette (Ca) into a promoter (P), a scar (Sc), a cistron(C)	Ca	P Sc C
6	Transform a cistron (C) into two cistrons (C) with a scar (Sc) in between	C	C Sc C
7	Transform a cistron (C) into a rbs (R) and a gene (G) with a scar (Sc) in between	C	R Sc G
8	Transform a terminator (T) into two terminators (T) with a scar (Sc) in between	T	T Sc T
9	Transform agene (G) into two genes (G) with a scar (Sc) in between	G	G Sc G

4　Mathematic model

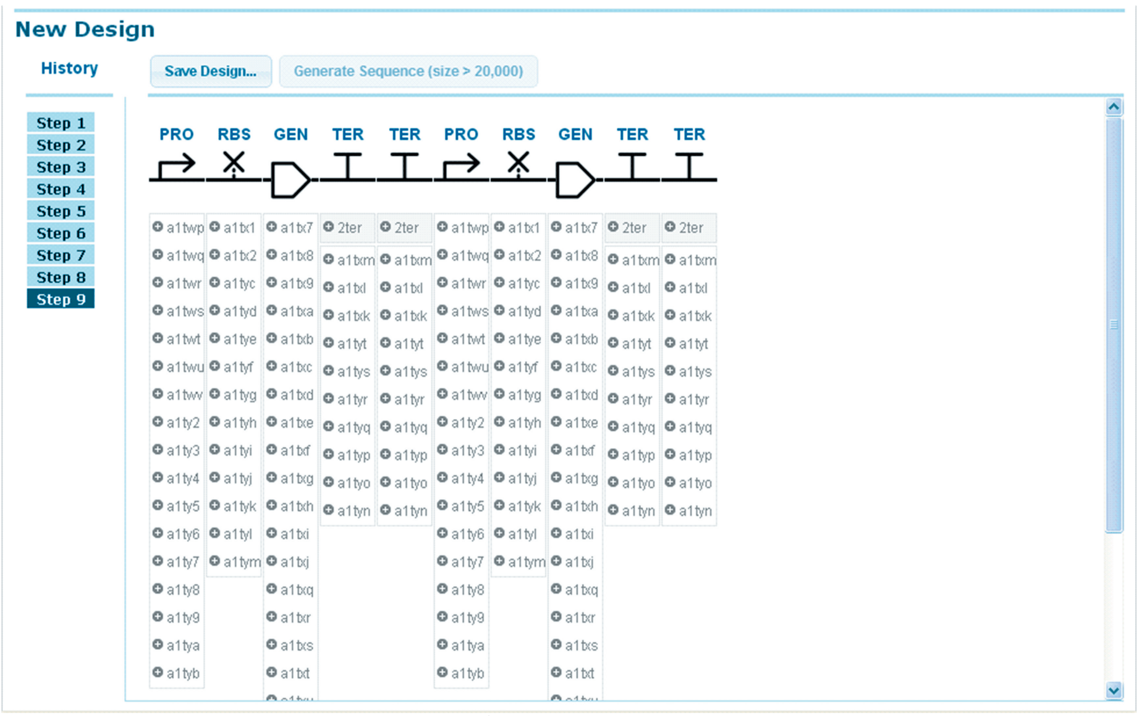
At the last step of GenoCAD design, every icon has its option. It is somewhat difficult for designer to choose the most suitable part to finish the design (Fig.1). To overcome this, statistical language model (SLM) is introduced. In this model, whether a sentence (S) is meaningful and reasonable is based on the probability it will happen. A sentence (S) is composed of a sequence of words. Here S is a genetic construct designed in GenoCAD and the words are imported basic parts. Now, S = $part_1, part_2, \cdots, part_n$ and we need to know its P(S) --- the probability it will happen.

$$P(S) = P(part_1, part_2, \ldots, part_n) \tag{1}$$

According to conditional probability formula

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

In formula 2, P($part_1$) means the probability $part_1$ appears in the design. P($part_2$ | $part_1$) means 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 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 not easy. And calculating P($part_n$ | $part_1$, $part_2$, $\cdots$, $part_{n-1}$) is very difficult, because much more variables are involved in. The conditions are too complex to gauge. Based on Markov Hypothesis we think the probability a part appear is only concerned with the part prior to it.



PRO	RBS	GEN	TER	TER	PRO	RBS	GEN	TER	TER
a1twp	a1tx1	a1tx7	2ter	2ter	a1twp	a1tx1	a1tx7	2ter	2ter
a1twq	a1tx2	a1tx8	a1txm	a1txm	a1twq	a1tx2	a1tx8	a1txm	a1txm
a1twr	a1tyc	a1tx9	a1txl	a1txl	a1twr	a1tyc	a1tx9	a1txl	a1txl
a1tws	a1tyd	a1txa	a1txk	a1txk	a1tws	a1tyd	a1txa	a1txk	a1txk
a1twt	a1tye	a1txb	a1tyt	a1tyt	a1twt	a1tye	a1txb	a1tyt	a1tyt
a1twu	a1tyf	a1txc	a1tys	a1tys	a1twu	a1tyf	a1txc	a1tys	a1tys
a1twv	a1tyg	a1txd	a1tyr	a1tyr	a1twv	a1tyg	a1txd	a1tyr	a1tyr
a1ty2	a1tyh	a1txe	a1tyq	a1tyq	a1ty2	a1tyh	a1txe	a1tyq	a1tyq
a1ty3	a1tyi	a1txf	a1typ	a1typ	a1ty3	a1tyi	a1txf	a1typ	a1typ
a1ty4	a1tyj	a1txg	a1tyo	a1tyo	a1ty4	a1tyj	a1txg	a1tyo	a1tyo
a1ty5	a1tyk	a1txh	a1tyn	a1tyn	a1ty5	a1tyk	a1txh	a1tyn	a1tyn
a1ty6	a1tyl	a1txi			a1ty6	a1tyl	a1txi		
a1ty7	a1tym	a1txj			a1ty7	a1tym	a1txj		
a1ty8		a1txq			a1ty8		a1txq		
a1ty9		a1txr			a1ty9		a1txr		
a1tya		a1txs			a1tya		a1txs		
a1tyb		a1txt			a1tyb		a1txt		

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

So formula 2 can be simplified as

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

Now P(S) ---the probability a S will happen can be calculated. Formula 3 is the Bigram Model of statistical language model. Then according to conditional probability formula

$$P(part_i | part_{i-1}) = \frac{P(part_{i-1}, part_i)}{P(part_{i-1})} \quad (4)$$

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

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

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

$$\text{And } P(part_i | part_{i-1}) \approx \frac{Count(part_{i-1}, part_i)}{Count(part_{i-1})} \quad (5)$$

In this way any component in formula 3 can be calculated.

At the last step of design (Fig. 1), there are too many combinations of basic parts to finish the design. Which one is the most reasonable and meaningful? We think the one with the largest appearing probability is. We have all the candidate paths, and a path will result a S (a path = a S = $part_1, part_2, \cdots, part_n$). The best path is represented with PATH.

$$PATH = \underset{all\_S}{\operatorname{argmax}}(P(S))$$

To avoid memory overflow when performing algorithm in computer, we take the log of P(S).

$$\begin{aligned} PATH &= \underset{all\_S}{\operatorname{argmax}}(\log P(S)) \\ &= \underset{all\_S}{\operatorname{argmax}}\Big(\log\Big(P(part_1) \times \prod_{i=2}^{n} P(part_i | part_{i-1})\Big)\Big) \\ &= \underset{all\_S}{\operatorname{argmax}}\Big(\log P(part_1) + \sum_{i=2}^{n} \log P(part_i | part_{i-1})\Big) \end{aligned} \quad (6)$$

According to formula 5, we got formula 7 and 8

$$P(part_i | part_{i-1}) = \frac{Count(part_{i-1}, part_i)}{Count(part_{i-1})} \quad (7)$$

$$P(part_1) = \frac{Count(part_1)}{Count(all\_parts)} \quad (8)$$

Because we extracted the dataset from a relatively sparse corpus, the zero-frequency problem would arise when parts pair never occurred in the training corpus. To overcome this, we use Add-one (Laplace) Smoothing[17]. So formula 7 should be

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

In formula 9 N is the number of Bigram (parts pair). Formula 8 and 9 will be used to fill the corresponding component in formula 6. The resulted *PATH* will be the S with the largest appearing probability in all candidate paths. And we will use dynamic programming algorithm to select the *PATH* from all candidates.

5 Algorithm

Now we need to find a path in the lattice in Fig. 1. This path is composed of series of parts and will be the *S* with the largest probability. It means how we can solve formula 6. The algorithm originates from the Viterbi algorithm[18] and will consist of three steps.

Firstly, build a candidate lattice. 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 were added respectively (Fig. 2). Every node is a triple-tuple < name, *V*, *P*>, and the first element name was filled with basic part name.

Secondly, fill the lattice. 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* will be filled with the maximum value selected from combining operation of two nodes in adjacent columns. *P* will store the address of the node prior to it where *V* comes from via combining operation.

1) The first column, for the B node let $V = 0$ and $P =$ NULL.

2) The second column, every node < name, *V*, *P* > (name ∈ {I0500, R0011, ⋯ , R0040, ⋯}) will combine with *B* node and calculate its *V* and *P*.

$$V = V_B + \log P(part) = \log P(part)$$

$$P = address\_of\_B$$

3) The third column, every node <name, V, P > (name ∈ {R0032, R0034, ⋯ , R0041, ⋯}) will combine with every node in the second column and calculate its *V* and *P*.

$$V = \max\{V_{prior} + \log P(part \mid part_{prior})\}$$

$$P = address\_where\_V\_comes\_from$$

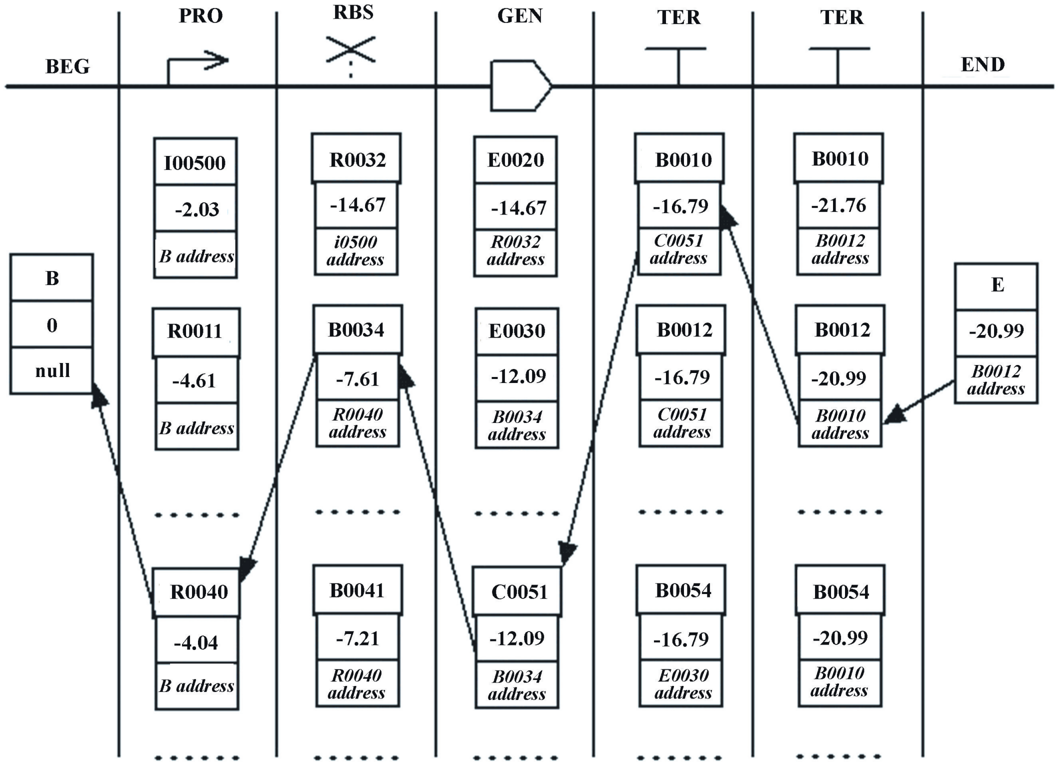


Fig. 2 The process of building lattice, filling lattice, and recalling

4) Repeat 3), every node in current column will combine with every node in prior column and calculate its *V* and *P*.

5) In the END column, *V* is the maximum value selected from the nodes in prior column, *P* will store the address of the node where *V* comes form.

Thirdly, recall and get thePATH. Start from node E, search the P prior to it continually (Fig. 2).

Finally, the PATH with the largest probability will be found and the resulted S is the optimized genetic construct. If the length of S is L and the maximum node number in a column is D, the algorithm complexity of this algorithm will be $O(L \cdot D^2)$, and the algorithm complexity of exhaustive algorithm is $O(D^L)$.

6 Results

To demonstrate how to optimize a GenoCAD design, 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 *BAT*2 and *THI*3 genes produces isoamyl alcohol, and the second one catalyzes the conversion of the cellular metabolite leucine to isoamyl acetate or banana odor. We design the system according to the grammar described previously. At the last step, we need to choose suitable basic parts to finish the design (Fig. 3). After the genes we want to express are determined, the optimizing algorithm will be implemented by a Perl script. At the first round of implementing algorithm, it recommended the parts series R0040–B0034–J45008–B0030–J45009–R0040–B0034–J45014–B0010–B0012. At the second round, when we excluded RBS B0034, the algorithm recommended the series R0040–B0030–J45008–B0030–J45009–R0040–B0030–J45014–B0010–B0012. At the third round, when we excluded promoter R0040 in the first column, the algorithm recommended the series R0011–B0030–J45008–B0030–J45009–R0040–B0030–J45014–B0010–B0012. And this is the real parts what banana odor biosynthetic system consists of. When we develop new project and carry out the algorithm at the last step, the algorithm will give out an optimized result based on experience. If we need some other options, we can exclude some parts and repeat the algorithm. It will recommend some other optimized results for consideration. If we have known that some parts are definitely connected, we can determine them first then implement the algorithm.

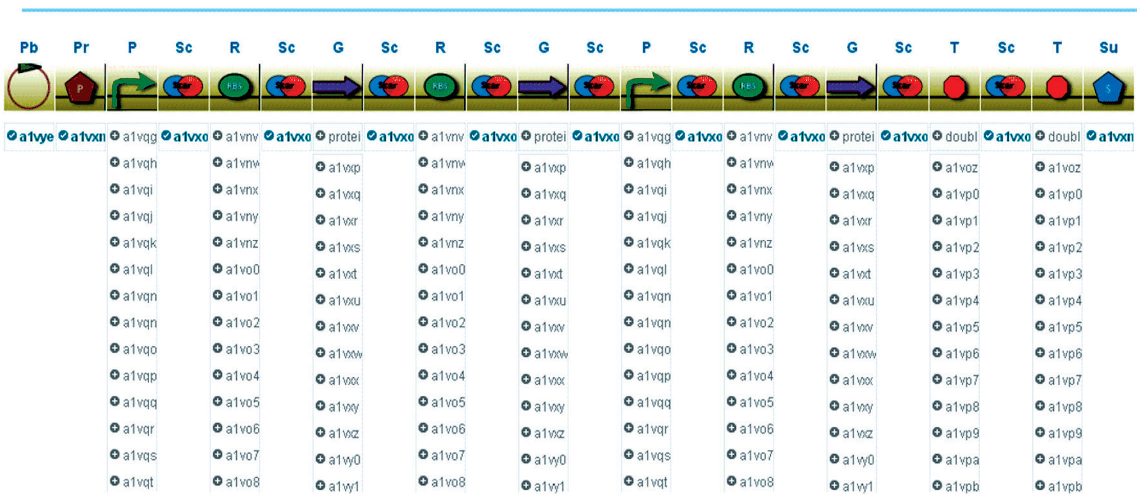


Fig. 3 The last step of designing the banana odor biosynthetic system

7 Discussion

This article has presented a statistical language model for synthetic biological parts assembling. After converting the BioBrick parts assembly process into a Bigram Model, a dynamic programming algorithm can be carried out to select an optimized result. The algorithm can be iterated then gives out different optimized results for consideration, but it can still not be embedded into other software. The method can be not only used to optimize a design in a synthetic biological robotic platform such as GenoCAD, but also independently used to automate the DNA assembly process in synthetic biology. After inputting categories of synthetic biological parts according to a grammar, the algorithm automatically assemble suitable parts to form a reasonable construct based on experience. In this way, redundant operations can be reduced and the time and cost required for conducting biological experiment ought to be minimized. Compared with other methods[12], the algorithm complexity of this method is $O(L \cdot D^2)$. It doesn't possess the advantage of speed but it can select the most suitable parts to form a bio-system referred to other successful cases, but it is better than the algorithm complexity of exhaustive algorithm which is $O(D^L)$. As described previously, this method is based on Bigram Model. It means every part involved in the assembly process is only concerned with the part 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 RBS but also its promoter. In order to simulate real world assembly process, N-gram Model should be introduced. This model means every part involved in the assembly process is concerned with N-1 parts prior to it. But in this model the conditional probability is very difficult to calculate. When $N = 3$ or 4, 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]. Next step we will develop a 3-gram Model and take plasmids backbone sequence into account to facilitate the DNA assembly process in synthetic biology.

When calculating the conditional probability, we used Add-one (Laplace) smoothing to overcome zero-frequency problem. It is always not a good choice [17]. It was used for considering that any two parts compliant with the same standard can be connected and for simplicity. Due to few applications of statistical language model in synthetic biological informatics, we do not know which smoothing technology is more effective. But the weakness of Add-one (Laplace) smoothing such as giving too much of the probability space to unseen events, worst at predicting the actual probabilities of bigrams are well-known[17]. We will develop 3 or 4-gram Model and expand the corpus to simulate the assembly process more reasonably. And other smoothing technology such as Good-Turing Smoothing, Katz backoff, Interpolation Smoothing [19-20] will be considered and used to improve the mathematic model. As described previously, we downloaded a relatively sparse corpus from iGEM website. We consider expanding the corpus to widely used plasmids and count the usage of features and two or three adjacent features. In this way, the statistical language model can be used more universally and tested in synthetic biology. But the description of the nature of parts is a more difficult issue. This can be solved by the development of an ontology giving the community a common controlled vocabulary to describe genetic parts. And developing the Synthetic Biology Open Language will promote this process.

Acknowledgements

Authors acknowledgethe BBF and the iGEM community for contributing the RFCs and BioBricks parts upon which this project was developed. Thanks for Prof. Peccoud and Mandy Wilson from Virginia Bioinformatics Institute guiding authors collecting the data.

参考文献(References)

[1] GOLER J A, BRAMLETT B W, PECCOUD J. Genetic design: rising above the sequence[J]. Trends Biotechnology, 2008(26):538-544.

[2] GRASLUND S, NORDLUND P, WEIGELT J, et al. Protein production and purification[J]. Nature Methods, 2008, 5(2):135-146.

[3] GHAEMMAGHAMI S, HUH W K, BOWER K, et al. Global analysis of protein expression in yeast[J]. Nature, 2003(425): 737-741.

[4] CZAR M J, CAI Y, PECCOUD J. Writing DNA with GenoCAD[J]. Nucleic Acids Research, 2009, 37(suppl 2): W40-W47.

[5] CAI Y, WILSON M L, PECCOUD J. GenoCAD for iGEM: a grammatical approach to the design of standard-compliant constructs[J]. Nucleic Acids Research, 2010, 38(8): 2637-2644.

[6] ADAMES N R, WILSON M L, FANG G, et al. GenoLIB: a database of biological parts derived from a library of common plasmid features[J]. Nucleic Acids Research, 2015, 43(10): 4823-4832.

[7] ISAACS F J, DWYER D J, DING C, et al. Engineered riboregulators enable posttranscriptional control of gene expression[J]. Nature Biotechnology, 2004(22): 841-847. DOI: 10.1038/nbt986.

[8] GARDNER T S, CANTOR C R, COLLINS J J. Construction of a genetic toggle switch in *Escherichia coli*[J]. Nature, 2000, 403(6767), 339-342.

[9] Atkinson M R, Savageau M A, Myers J T, et al. Development of genetic circuitry exhibiting toggle switch or oscilla-

tory behavior in *Escherichia coli*[J]. Cell, 2003(113):597-607.

[10] ARKIN A. Setting the standard in synthetic biology[J]. Nature Biotechnology, 2008(26):771-774.

[11] CANTON B, LABNO A, ENDY D. Refinement and standardization of synthetic biological parts and devices[J]. Nature Biotechnology, 2008, 26(7): 787-793. DOI: 10.1038/nbt0708-771.

[12] DENSMORE D, HSIAU T H C, BATTEN C, et al. Algorithms for automated DNA assembly[J]. Nucleic Acids Research, 2010, 38(8):2607-2616. DOI: 10.1093/nar/gkq165.

[13] COLL A, WILSON M L, GRUDEN K, et al. Rule-based design of plant expression vectors using GenoCAD[J]. PLoS ONE, 2015, 10(7): e0132502. DOI: 10.1371/journal.pone.0132502.

[14] JELINEK F. Statistical methods for speech recognition (Language, Speech, and Communication)[M]. Cambridge, MA: MIT Press, 1998.

[15] PHILLIPS I E, SLIVER P A. A new biobrick assembly strategy sesigned for facile protein engineering[J/OL]. DSpace@MIT, http://dspace.mit.edu/handle/1721.1/32535, 2006-04-20/2016-04-01.

[16] CAI Y, HARTNETT B, GUSTAFSSON C, et al. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts[J]. Bioinformatics, 2007, 23(20): 2760-2767. DOI: 10.1093/bioinformatics/btm446.

[17] CHEN S F, GOODMAN G. An empirical study of smoothing techniques for language modeling[J]. Computer Speech and Language, 1999(13): 359-394.

[18] VITERBI A J. A personal history of the viterbi algorithm [J]. IEEE Signal Processing Magazine, 2006(23): 120-142.

[19] HUANG F L, YU M S, HWANG C Y. An empirical study of good-turing smoothing for language models on different size corpora of chinese[J]. Journal of Computer and Communications, 2013(1): 14-19.

[20] Katz S M. Estimation of probabilities from sparse data for the language model component of a speech recogniser[J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1987, 35(3):400-401.