SCFGs para Análise de Sequências de RNA

Heitor Baldo*
Algoritmos em Bioinformática (IBI5037)
Resenha

2º Semestre, 2019

Resumo

Nesta resenha, discorremos sobre SCFGs para análise e previsão de estruturas secundárias de RNA. Apresentamos brevemente a teoria clássica das SCFGs, incluindo alguns exemplos e os principais algoritmos. Discutimos sobre o problema da ambiguidade de gramáticas, sobre métodos de design de gramáticas e, finalmente, sobre suas aplicações para previsão de estruturas secundárias de RNA.

Conteúdo

*NUSP: 11571801 / hbaldo@usp.br

1 Introdução

A previsão da estrutura secundária de RNA é um desafio de longa data no campo da bioinformática. O problema consiste em derivar uma estrutura bidimensional a partir da sequência primária (sequência de nucleotídeos) de RNA. Com isto queremos prever a posição das ligações de hidrogênio em uma molécula de RNA com base apenas em sua sequência primária. Essas previsões podem ser usadas para entender melhor o funcionamento das células, as características da expressão gênica e os mecanismos envolvidos na produção de proteínas.

Diversos métodos computacionais foram desenvolvidos com o objetivo de resolver esse problema, mas ainda não foi possível desenvolver um método ótimo. O problema se torna ainda mais computacionalmente difícil ao permitir a ocorrência de um certo tipo de subestrutura, a saber, os *pseudonós*, que ocorrem nas estruturas reais de RNA e, portanto, são relevantes na modelagem computacional.

O foco principal desta resenha consiste na análise dos artigos [1], [2] e [6], acentando-se nas *Gramáticas Estocásticas Livres de Contexto* (comumente abreviada por "SCFGs", do inglês *Stochastic Contex-Free Grammars*, e às vezes grafada como *Probabilistic Context Free Grammars*), e como elas podem ser usadas para fornecer uma abordagem probabilística à previsão de estruturas secundárias de RNA. Discorreremos brevemente sobre a teoria por trás das SCFGs, incluindo os principais algoritmos utilizados, sobre ambiguidades de gramáticas, métodos para obtenção dessas gramáticas e sobre suas aplicações na análise de sequências de RNA.

2 Gramáticas Estocásticas Livre de Contexto

Antes de definirmos formalmente uma SCFG, vamos definir formalmente uma *Gramática Livre de Contexto* (comumente abreviada como "CFG", do inglês *Contex-Free Grammar*).

2.1 Gramáticas Livre de Contexto (CFGs)

Definição 2.1. Uma CFG G é uma 4-tupla (V, T, P, S) em que:
(i) V é um conjunto finito de não-terminais ("estados");
(ii) T é um conjunto finito de terminais;
(iii) P é um conjunto finito de regras de produção;
(iv) S é o não-terminal inicial ($S \in V$).

Exemplo 2.1. A seguinte CFG gera estruturas de RNA: $V = \{s\}$, $T = \{A, C, U, G\}$ e

$$P = \begin{cases} s \to AsU \mid UsA \mid CsG \mid GsC \\ s \to As \mid Cs \mid Gs \mid Us \\ s \to sA \mid sC \mid sG \mid sU \\ s \to ss \\ s \to \epsilon \end{cases}$$

Exemplo 2.2. Outro exemplo simples de CFG que gera estruturas de RNA: $V = \{S_0, S_1, S_2, ..., S_{13}\}$, $T = \{A, C, U, G\}$, com um conjunto de regras de produção P como mostrado a seguir:

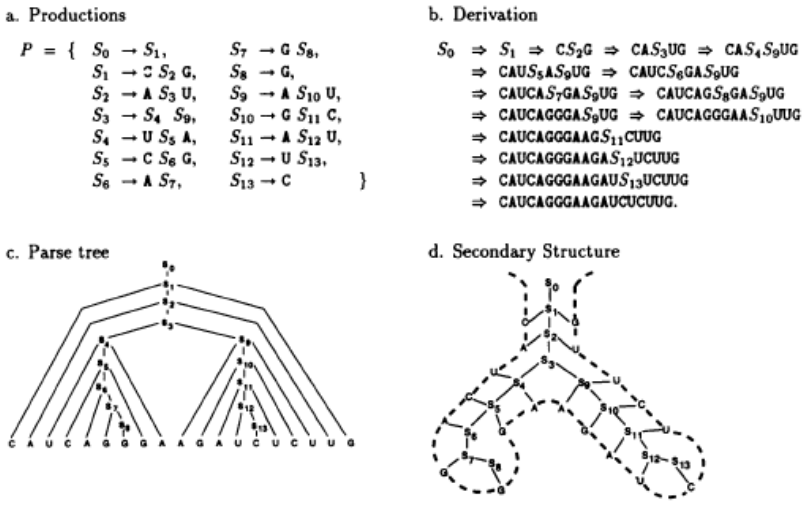

Figura 1. Uma CFG que pode produzir um conjunto de moléculas de RNA. Uma *derivação* de CFG (b) tem uma representação gráfica conhecida como *árvore de derivação* (c). A estrutura secundária obtida da derivação (b) (sem pseudonós) é mostrada em (d) (figura retirada de [6]).

Observação 2.1. CFGs não são úteis para predição de estruturas, pois, dada uma sequência de RNA, pode haver um grande número de árvores de derivação válidas, cada uma correspondendo a uma possível estrutura secundária de RNA (elas são ambíguas). Para tentar contornar este tipo de problema, utilizamos as SCFGs.

2.2 Gramáticas Estocásticas Livre de Contexto (SCFGs)

Vamos agora definir formalmente uma Gramática Estocástica Livre de Contexto (SCFG).

Definição 2.2. Uma SCFG é uma CFG $G = (V, T, P, S)$, tal que o conjunto de regras de produção P possui uma distribuição de probabilidades. Ou seja, uma SCFG especifica uma probabilidade para cada produção na gramática.

Referimo-nos ao conjunto total de probabilidades (os *parâmetro* do modelo) por θ. Podemos denotar, assim, uma SCFG por (G, θ).

Exemplo 2.3. Uma SCFG simples é definida como segue: $V = \{S, w_1, w_2, w_3\}$, $T = \{A, C, U, G\}$,

$$P = \begin{cases} S \to Aw_1U^{0.25} \mid Cw_1G^{0.25} \mid Gw_1C^{0.25} \mid Uw_1A^{0.25} \\ w_1 \to Aw_2U^{0.25} \mid Cw_2G^{0.25} \mid Gw_2C^{0.25} \mid Uw_2A^{0.25} \\ w_2 \to Aw_3U^{0.2} \mid Cw_3G^{0.3} \mid Gw_3C^{0.3} \mid Uw_3A^{0.2} \\ w_3 \to GAAA^{0.2} \mid GCAA^{0.8} \end{cases}$$

Os números superescritos representam as probabilidades associadas a cada produção. Observe que a soma das probabilidades associadas às produções são sempre iguais a 1.

Observação 2.2. Por definição, uma SCFG descreve uma distribução de probabilidade conjunta $P(x, \pi|G, \theta)$, dado uma sequências x, sobre todas possíveis árvores de derivação π, ou seja, $P(x, \pi|G, \theta)$ é o produto de todas as probabilidades das regras de produção usadas numa árvore de derivação π para uma sequência x.

2.3 Forma Normal de Chomsky

Para desenvolver algoritmos para analisar sequências através de modelos de gramáticas, é conveniente restringi-lás a uma determinada *forma normal* (que possui apenas alguns poucos tipos de produções). A forma normal mais comumente usada é a *Formal Normal de Chomsky* (CNF), definida formalmente a seguir.

Definição 2.3. Dizemos que uma CFG está na CNF se todas as suas regras de produções são da forma $A \to YZ$ ou $A \to a$ ou $S \to \epsilon$, em que A, S, Y, Z são não-terminais, a é um terminal e ϵ é a string vazia.

Qualquer CFG pode ser posta nessa forma normal, ou seja, toda CFG é equivalente a uma gramática na CNF, e, provavelmente, toda SCFG pode ser convertida a um conjunto de regras desse tipo.

Observação 2.3. A despeito das CNFs serem úteis, uma gramática na CNF não pode introduzir nucleotídeos pareados a partir de uma única produção e, por conseguinte, não captura a estrutura do RNA de forma direta. Assim, em aplicações, é conveniente evitar essa conversão. No entanto, uma nova forma normal foi construída em [1], de tal forma que podusse capturar as características fundamentais da estrutura secundária do RNA: ramificação, bases não-pareadas e bases pareadas. Essa forma normal é definida como segue.

Definição 2.4. Para não-terminais T, U, V, a *forma normal de emissão dupla*[1] possui apenas regras da forma $T \to UV$ ou $T \to .$ ou $T \to (U)$, em que $(,)$ representa pares de bases e $.$ representa a emissão nula.

Essa forma normal será utilizada na abordagem evolucionária para busca de gramáticas, como veremos adiante.

2.4 Algoritmos para SCFGs

Em geral, quando trabalhamos com SCFGs, utilizamos algoritmos para os três principais propósitos:

1) **Árvore de derivação ótima:** Um algoritmo de programação dinâmica para SCFGs que pode determinar a árvore de derivação mais provável para uma sequência x com complexidade temporal proponcional a L^3, sendo L o tamanho de x, é o *algoritmo CYK* (Cocke-Younger-Kasami).

2) **Probabilidade da Sequência:** O *algoritmo Inside* é um algoritmo de programação dinâmica (análogo ao algoritmo *forward* para HMMs) que é usado para obter a probabilidade total de derivação de uma sequência x, dado o modelo (G, θ), somando sobre todas as possíveis árvores de derivação.

[1] Assim denominada porque bases pareadas são emitidas simultaneamente.

3) **Estimação dos Parâmetros:** Um dos algoritmos mais comumentes usados para estimação dos parâmetros de uma SCFG (ou seja, as probabilidades de produção), a partir de um conjunto de sequências de treino, é o *algoritmo Inside-Outside* (IO). Assim como o algoritmo *forward-backward* para HMMs, este algoritmo é um método de Maximização de Expectativa (EM) para obter a maior probabilidade dos parâmetros da SCFG.

2.4.1 O Algoritmo CYK

O algoritmo CYK calcula a árvore de derivação (e por conseguinte a estrutura secundária) mais provável para uma sequência x (de comprimento L), dado uma SCFG (G, θ). Para uma SCFG, o algoritmo CYK é caracterizado pelas seguintes etapas:

Inicialização: $\gamma(i, i-1) = log\ p(S \to \epsilon)$

Iteração:

$$\gamma(i,j) = max \begin{cases} \gamma(i+1, j-1) + log\ p(S \to x_i S x_j) \\ \gamma(i+1, j) + log\ p(S \to x_i S) \\ \gamma(i, j-1) + log\ p(S \to S x_j) \\ max_{i<k<j}\{\gamma(i,k) + \gamma(k+1,j) + log\ p(S \to SS)\} \end{cases}$$

Terminação: $\gamma(1, L) = log\ P(x, \hat{\pi}|G, \theta)$

Assim, o algoritmo CYK encontra uma árvore de derivação ótima $\hat{\pi}$ para a sequência x fazendo o traceback:

$$\hat{\pi} = arg\ max_{\pi}\ P(x, \pi|G, \theta).$$

2.4.2 Estimação de Parâmetros

Podemos aprender os parâmetros de uma SCFG a partir de sequências de treinamento usando uma abordagem de EM chamada *algoritmo Inside-Outside* (IO) [5]. Esse algoritmo pode ser visto como uma generalização para SCFGs do *algoritmo forward-backward* para estimativa de parâmetros em HMMs.

Todavia, em [6] foi utilizado um algoritmo alternativo, chamado *Tree-Grammar EM training algorithm*, para estimar os parâmetros de uma SCFG a partir de sequências não-alinhadas de RNA de treinamento. Depois de projetar uma gramática inicial adequada, as sequências de treinamento são empregadas apenas para refinar as estimativas das probabilidades de produção da gramática.

Em [2], os parâmetros de cada SCFG foram estimados a partir de frequências observadas em estruturas secundárias anotadas. O conjunto de treinamento foi composto por subunidades grandes e pequenas de rRNA obtidas do *European Ribosomal RNA Database*[2].

Já em [1], foi utilizado tanto o algoritmo CYK[3] quanto o algoritmo Inside-Outside, para o treinamento das gramáticas encontradas pela abordagem evolucionária.

[2] http://bioinformatics.psb.ugent.be/webtools/rRNA/

[3] Para o algoritmo CYK, no caso de gramáticas ambíguas, não se pode saber qual derivação produziu a estrutura conhecida; portanto, as probabilidades não podem ser obtidas. Consequentemente, treinamos essas gramáticas selecionando aleatoriamente uma derivação.

2.5 Ambiguidade de Gramáticas

Definição 2.5. Uma gramática é dita ser *ambígua* quando existe mais do que uma árvore de derivação possível para alguma sequência. Ademais, quando múltiplas árvores de derivação descrevem a mesma estrutura secundária, chamamos a gramática de *estruturalmente ambígua*.

Exemplo 2.4. A gramática a seguir é ambígua: $V = \{S\}$, $T = \{a, c, g\}$,

$$P = \begin{cases} 1.\ S \to gSc \\ 2.\ S \to gS \\ 3.\ S \to aa \end{cases}$$

Com esta gramática temos três possíveis árvores de derivação que geram a string $GGGAACC$, a saber, pela aplicação das regras $2, 1, 1, 3$ e $1, 2, 1, 3$ e $1, 1, 2, 3$.

Observação 2.4. Se uma estrutura A tem uma derivação com probabilidade 0.3 e uma estrutura B tem duas derivações com probabilidade 0.25 cada, o algoritmo CYK irá escolher a estrutura A, a despeito da estrutura B ser mais provável. Ou seja, ambiguidade pode reduzir a qualidade das predições feitas com o algoritmo CYK.

Ambiguidade de gramáticas é um problema na análise de sequências de RNA. Por exemplo, se uma gramática é estruturalmente ambígua, não podemos igualar a probabilidade de uma árvore de derivação com a probabilidade de sua estrutura. Assim, uma estrutura ótima não pode ser eficientemente obtida pelo algoritmo CYK, forçando-nos ou a utilizarmos gramáticas estruturalemente inequívocas (não ambíguas), ou teremos que assumir que é uma aproximação válida assumir que uma árvore de derivação ótima nos fornece a estrutura ideal.

A ambiguidade é um dos motivos pelos quais queremos desenvolver métodos de busca de gramáticas que sejam boas o suficiente, como veremos a seguir.

3 Encontrando Gramáticas

Primeiramente, seria natural a busca por gramáticas pelo método da força bruta. Nesse caso, teríamos de construir e avaliar a eficácia de gramáticas relativamente pequenas. Isso foi feito em [2], e averiguaremos os resultados obtidos a seguir.

3.1 Força Bruta

Em [2] foram analisadas nove gramáticas construídas a mão. Duas delas mostraram-se ambíguas (a saber, $G1$ e $G2$), e as outras sete mostraram-se inequívocas.[4] A seguir, vamos analisar as gramáticas $G3, G4, G5$ e $G6$.

Exemplo 3.1. Para terminais $a, \hat{a}$ e não-terminais S, L, R, F, temos quatro gramáticas inequívocas:

[4]Por simplicidade, as quatro gramáticas G2, G7, G8 e G9, não foram consideradas neste texto pois envolvem produções do tipo $p^{b\hat{b}} \to aP^{a\hat{a}}\hat{a}$, em que a probabilidade de emissão de um novo par $a, \hat{a}$ depende do par de bases anterior $b, \hat{b}$ ($a, \hat{a}, b, \hat{b}$ terminais, e $P^{a\hat{a}}, P^{b\hat{b}}$ não-terminais). A notação $P^{a\hat{a}}$ é utilizada para incluir parâmetros de empilhamento de pares de bases (stacking grammar).

$$G3 : \begin{cases} S \to aS\hat{a} \mid aL \mid Ra \mid LS \\ L \to aS\hat{a} \mid aL \\ R \to RA \mid \epsilon \end{cases}$$

$$G4 : \begin{cases} S \to aS \mid T \mid \epsilon \\ T \to Ta \mid aS\hat{a} \mid TaS\hat{a} \end{cases}$$

$$G5 : S \to aS \mid aS\hat{a}S \mid \epsilon$$

$$G6 : \begin{cases} S \to LS \mid L \\ L \to aF\hat{a} \mid a \\ F \to aF\hat{a} \mid LS \end{cases}$$

Cada uma das quatro gramáticas foi conjecturada como inequívoca por inspeção. Cada uma também passou no teste empírico de ambiguidade descrito em [2]. Consulte o mesmo artigo para exemplos de árvores de derivação destas gramáticas.

Observação 3.1. Cada uma dessas gramáticas impõe restrições ligeiramente diferentes às estruturas possíveis. $G3$ impõe um comprimento mínimo de *hairpin loop* de um nucleotídeo, $G6$ tem um mínimo de dois e $G4$ e $G5$ não impõem comprimentos mínimos de *hairpin loop*. Além disso, $G6$ não pode emitir uma string vazia, ϵ, enquanto $G3$, $G4$ e $G5$ podem.

A acuracidade de cada gramática para a predição da estrutura secundária foi medida pelo cálculo da sensibilidade e da precisão (PPV) sobre um conjunto de *benchmark* de 403 estruturas secundárias confiáveis derivadas de análise comparativa. A partir disso, constatou-se que a gramática $G5$ teve a pior performance na predição; a $G3$ obteve uma performance mediana, enquanto que a $G6$ obteve a melhor performance dentre as quatro gramáticas.

Ademais, as performances dessas gramáticas foram comparadas com às de alguns algoritmos que utilizam minimização de energia para predição da estrutura secundária, a saber, *mfold*, Vienna RNA, PKNOTS e RNAstructure, para o mesmo conjunto de *benchmark*. O resultado foi que todas as quatro gramáticas tiveram uma performance inferior aos desses algoritmos.

3.2 Abordagem Evolucionária

Como mencionamos anteriormente, em [2] foram analisadas nove gramáticas construídas a mão, e havia pouca motivação para a construção das suas regras de produção. Por isso, uma busca computacional em um espaço maior de gramáticas pode encontrar gramáticas melhores. Todavia, na busca exaustiva por gramática pequenas, problemas computacionais surgem quase que imediatamente. Em vista disso, devemos procurar por métodos computacionais eficientes. Um exemplo é o artigo [1], em que uma abordagem evolucionária foi empregada para busca de novas gramáticas.

Algoritmo Evolucionário

Devemos colocar nosso algoritmo evolutivo na *forma normal de emissão dupla* (apresentada na seção 2.3), para termos uma busca eficiente. Para essa forma normal, para m não-terminais,

existe $2^{m^3+m^2+m}$ gramáticas (m^3 regras de produções do tipo $T \to UV$, m^2 do tipo $T \to (U)$ e m do tipo $T \to .$).[5] A maneira como o algoritmo evolutivo pesquisa o espaço é determinada pela escolha da *população inicial* e dos métodos de *mutação*, *reprodução* e *seleção*. A seguir, descreveremos resumidamente o método evolucionário e os resultados obtidos em [1].

População Inicial: Começamos com uma população inicial de gramáticas pequenas e usamos regras de mutação e de reprodução para aumentar o número de não-terminais e de regras de produção. A população inicial foi composta por dezesseis gramáticas da forma:

$$S \to \left\{ \begin{array}{c|c|c|c|c|} SS & SB & BS & BB & (S) \\ - & - & - & - & \end{array} \right. .$$
$$B \to .$$

Mutação: As mutações constituem a maioria dos movimentos no espaço de pesquisa, portanto são particularmente importantes. Eles dão à gramática novas características, permitindo maior liberdade estrutural e acrescentam regras de produção que podem ser usadas imediatamente ou que podem permanecer inativas. Para não-terminais $V_i \in V$, e regras de produções correspondentes P_{V_i}, as mutações estocásticas permitidas foram:

(i) O não-terminal de início (e as regras de produção correspondente) muda;

(ii) Uma regra de produção é adicionada ou excluída;

(iii) Um novo não-terminal V' é adicionado juntamente com duas novas regras que garantem que V' seja alcançável e que $P_{V'}$ não seja vazia;

(iv) Um não-terminal é criado com regras idênticas a um já existente;

(v) Uma regra de produção da forma $V_i \to V_j V_k$ é alterada para $V_i \to V_j V_l$, ou $V_i \to V_l V_k$, ou $V_i \to V_l V_p$, ou a regra de produção da forma $V_i \to (V_j)$ é alterada para $V_i \to (V_k)$.

Reprodução: O modelo de reprodução produz uma gramática que pode produzir todas as derivações das gramáticas a partir das quais ela foi originada (gramáticas pais). A gramática G formado a partir da "reprodução"das gramáticas G_1 e G_2 (com símbolos iniciais S_1 e S_2) tem símbolo inicial S, não-terminais $V_1, ..., V_n$, $W_1, ..., W_n, B$, terminais $(,), .$ e regras de produções:

(i) $P_S = P_{S_1} \cup P_{S_2}$;

(ii) Para V_i: P_{V_i} são todas as ocorrências em que S_1 é substituído por S;

(iii) Para W_i: P_{W_i} são todas as ocorrências em que S_2 é substituído por S.

Seleção: Aumentamos a população em cada geração pela introdução de várias gramáticas recém-mutacionadas ou criadas, e, em seguida, reduzimos esse número a uma população de tamanho fixo por eliminação estocástica.

Nos experimentos, mais de 300.000 gramáticas foram pesquisadas. Foram encontradas várias gramáticas fortes usando ambos os algoritmos, CYK e IO. Das gramáticas encontradas, as gramáticas analisadas foram denotadas por $GG1, GG2, GG3, GG4$, $GG5$, $GG6$. Ademais, a gramática $KH99'$, que é o KH99 (Knudsen-Hein, 1999 [4]) na forma normal de emissão dupla, também foi incluída.

[5] Em vista disso, a busca por força bruta é viável computacionalmente para gramáticas com apenas dois não-terminais, além da regra $T \to .$ ($2^{14} = 16.384$ gramáticas), mas não para três não-terminais ($2^{39} = 549.755.813.888$ gramáticas).

Essas gramáticas tiveram valores de sensibilidade e VPP de aproximadamente $0,4 - 0,6$. Ainda, como esperado, gramáticas com estruturas semelhantes tiveram predições semelhantes. A diferença na performance entre KH99′ e KH99 foi pequena, e isso confirma que a representação de KH99 como KH99′ é boa.

As gramáticas $GG1 - GG6$ foram comparadas com métodos que utilizam princípios termodinâmicos, como o UNAfold e o RNAfold, e os resultados mostraram que esses métodos continuam tendo melhor performance que os métodos que utilizam SCFGs.

Como essas gramáticas são projetadas para prever a estrutura secundária do RNA usando o mesmo conjunto de treinamento, seria de esperar alguma semelhança nas previsões. Isso foi confirmado pelos resultados, sugerindo que novas gramáticas produzem diferentes tipos de estruturas, sendo assim boas representações de estruturas secundárias do RNA.

4 Aplicação de SCFGs para predição de estruturas secundárias de RNA

A maioria dos algoritmos para previsão de estruturas secundárias utilizam minimização de energia. Porém abordagens probabilísticas usando SCFGs podem ser usadas. O uso de SCFGs na predição de estruturas secundárias de RNA foi baseado no sucesso dos HMM para modelagem de proteínas e genes. Na realidade, as SCFGs são uma generalização dos HMMs.

A estrutura secundária pode ser predita através de dois métodos:

1) **Algoritmo CYK**: O algoritmo CYK determina, através de programação dinâmica, a probrabilidade da derivação mais provável, em seguida é feito o traceback para encontrar a estrutura mais provável.

2) **Posterior Decoding**: Pode-se empregar um método de *posterior decoding* usando matrizes de probabilidades de bases pareadas. Essas matrizes de probabilidade para uma SCFG são obtidas usando o algoritmo IO. Daí, a estrutura secundária com o número máximo esperado de posições corretas pode ser calculada através de programação dinâmica.

Há diversas abordagens que utilizam SCFG para predição de estruturas secundárias de RNA, como, por exemplo, os algoritmos Pfold e RNAalifold. O Pfold [3] é baseado no algoritmo KH99. Esse algoritmo utiliza uma SCFG para produzir uma distribuição de probabilidade inicial de estrutura de RNA[6].

Obviamente, a previsão da estrutura secundária de RNA é útil apenas para moléculas de RNA que são estruturadas, como as moléculas de tRNA, ao contrário dos mRNA, que são relativamente desestruturados. Em [6], SFCGs foram utilizadas para previsão de estruturas secundárias de tRNA.

Um dos pontos fracos das SCFGs para este tipo de previsão, é que elas não conseguem prever pseudonós.

[6] A saber, o Pfold utiliza a gramática $G6$ da seção anterior.

5 Considerações Finais

Neste breve trabalho, estudamos como o design de SCFGs impacta na predição de estruturas secundárias de RNA, como a ambiguidade estrutural de gramáticas pode ser deletéria na predição dessas estruturas e estudamos métodos para busca de gramáticas boas o suficiente que possam contornar, de certa forma, esses problemas.

O método da busca por força bruta mostrou-se computacionalmente ineficiente para gramáticas grandes e por isso consideramos o método evolucionário, que se mostrou relativamente eficiente na busca por gramáticas mais fortes. Não obstante, a precisão da previsão de estruturas secundárias por SCFGs mostrou-se ligeiramente inferior ao dos métodos que utilizam princípios termodinâmicos, como minimização de energia.

Referências

[1] Anderson, J., Tataru, P., Staines, J. et al., *Evolving stochastic context-free grammars for RNA secondary structure prediction.* BMC Bioinformatics (2012) 13, 78.

[2] Dowell, R. D. and Eddy, S. R. *Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction.* BMC Bioinformatics (2004) 5:71.

[3] Knudsen, B. and Hein, J. *Pfold: RNA secondary structure prediction using stochastic context–free grammars.* Nuc Acid Res (2003) 31: 3423–3428.

[4] Knudsen, B. and Hein, J. *RNA secondary structure prediction using stochastic context-free grammars and evolutionary history.* Bioinformatics (1999) 15, 446–454.

[5] Lari K. and Young S., *The estimation of stochastic context–free grammars using the inside–outside algorithm.* Comput Speech Language (1990) 4:35–56.

[6] Sakakibara Y., Brown M., Hughey R., Mian I. S., Sjölander K., Underwood R. C., Haussler D., *Stochastic context-free grammars for tRNA modeling.* Nucleic Acids Res. (1994) 22(23):5112-20.