Prediksi Struktur Sekunder RNA Menggunakan *Stochastic Context Free Grammar* dan *Grammatical Evolution*

Asriyanti Indah Pratiwi
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
asriyantiindahpratiwi@gmail.com

Agung Toto Wibowo, ST., MT.
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
atwbox@gmail.com

Gia Septiana W., S.Si., M.Sc.
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
giaseptiana@gmail.com

Salah satu metode prediksi struktur sekunder RNA (*ribonucleic acid)* adalah SCFG *(stochastic context free grammar)*. Namun SCFG memiliki ketegantungan yang tinggi terhadap *grammar*. *Grammar* yang kurang baik akan berdampak buruk terhadap performansi prediksi. Hal ini menyebabkan hasil prediksi menjadi tidak optimal. Oleh karena itu, penelitian ini berfokus dalam perancangan probabilitas setiap *production rules* dari *grammar* untuk meningkatkan nilai *sensitivity* dari *grammar* Watson Crick yang biasa digunakan dalam SCFG. Untuk mencapai nilai *sensitivity* yang lebih baik maka dalam penelitian ini dibangun sebuah sistem menggunakan *grammatical evolution* untuk mendapatkan probabilitas setiap *production rule* dari *grammar*. Penelitian ini berhasil meningkatkan nilai *sensitivity grammar*. Dari hasil pengujian didapatkan nilai *sensitivity* sebesar 0,32-0,42.

Keywords: prediksi, struktur sekunder RNA, SCFG, Grammatical Evolution , nilai sensitivity

I. PENDAHULUAN

Ribonucleic Acid atau yang sering disebut dengan RNA secara hirarki dapat dibagi ke dalam tiga level struktural yaitu struktur primer, struktur sekunder dan struktur tersier. Prediksi struktur sekunder RNA dari struktur primernya yang berupa urutan nukleatida sangat dibutuhkan oleh para ilmuwan biologi untuk mengetahui interaksi antar makromolekul pada RNA. Hal ini dikarenakan dengan mengetahui struktur sekunder suatu RNA, fungsionalitas RNA tersebut dapat diketahui sehingga para ilmuwan biologi dapat menggunakan informasi tersebut sebagai acuan dalam perancangan obat-obatan.

Beberapa metode untuk memprediksi struktur sekunder RNA pun bermunculan seperti *Bayesian Algorithm* [14], *Conditional Random Fields* [8] dan beberapa metode perbaikannya seperti *Chain Graph Model* [10], *Segmentation CRF* [9] serta *Hidden-Unit CRF* [11]. Selain itu terdapat beberapa metode lainnya seperti pengunaan *Neural Network* yang dikombinasikan dengan *Hidden Markov Model* [12][5][2].

Salah satu metode yang lazim digunakan ialah *Stochastic Contex-Free Grammar* atau yang seringkali disebut dengan SCFG. Penelitian terakhir mengenai SCFG adalah melakukan percobaan untuk mendapatkan probabilitas dari serangkaian *grammar* dengan melakukan evolusi dari estimasi frekuensi basa yang berpasangan dan tidak berpasangan dalam RNA. Selain itu, kita dapat menggunakan algoritma evolusi lainnya seperti *Grammatical Evolution* (GE) untuk mendapatkan probabilitas untuk setiap *production rule* dari sebuah *grammar*.

Tujuan dari penelitian ini adalah membangun suatu modul yang dapat memberikan probabilitas untuk setiap *production rule* dari *grammar* Watson, dimana probabilitas tersebut merupakan probabilitas yang membuat *grammar* Watson Crick memiliki *sensitivity* yang cukup baik. Modul ini dibangun menggunakan *Grammatical Evolution* dan SCFG.

Grammatical Evolution akan membangkitkan *grammar* Watson Crick dengan memungkinkan kemunculan *production rules* yang berulang. Kemunculan suatu *production rules*\ akan dihitung sebagai peluang digunakannya *production rules* pada saat pengimplementasian SCFG.

II. PENELITIAN TERKAIT

Stochastic Contex-Free Grammar atau yang seringkali disebut dengan SCFG. SCFG diimplementasikan dalam prediksi struktur sekunder RNA [1,4,6,7],karena kemampuannya mengikuti aturan sebuah *grammar* yang telah ditentukan sebelumnya untuk dapat memprediksikan struktur sekunder dari RNA. Sampai saat ini *grammar* yang lazim digunakan ialah *grammar* Watson Crick dengan probabilitas *grammar* yang terdistribusi *uniform*. Namun, grammar tersebut belum dapat dikatakan memuaskan karena SCFG yang menggunakan *grammar* tersebut memiliki nilai *sensitivity* dan *specificity* yang masih relatif rendah. Sehingga dewasa ini, penelitian mengenai SCFG berfokus pada bagaimana caranya menemukan probabilitas yang tepat untuk setiap *production rule* dari suatu *grammar*. Hal ini dikarenakan probabilitas yang berbeda akan menghasilkan struktur sekunder yang berbeda.

III. METODE

A. Stochastic Context Free Grammar (SCFG)

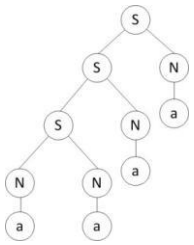
Suatu *Stochastic Context Free Grammar* (SCFG) dapat dinotasikan dengan G yang memiliki 4 Tuple (N, V, P, S), terdiri dari variabel non-terminal sejumlah N, variabel terminal sejumlah V, aturan produksi dengan probabilitas

penurunannya sejumlah P dan dimulai dari variabel non-terminal S[18]. Setiap aturan produksi dapat merubah satu variabel non-terminal dengan string non-terminal dan atau terminal lainnya [8]. Berikut merupakan contoh dari *stochastic contex free grammar* :

N->a (0.25)	N->u (0.25)
N->c (0.25)	N->g (0.25)
S->SN (0.5)	S->NN (0.2)
S->SS (0.3)	

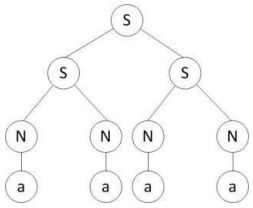
Gambar 1 Contoh Stochastic Context Free Grammar

Grammar diatas merupakan salah satu contoh *grammar* sederhana *rule production* S→SN memiliki peluang 50% untuk terjadi, aturan S→SS memiliki peluang kemunculan hanya sebesar 30%. Peluang muncul dari string seperti berikut S≈ SN ≈ SNN ≈ NNNN ≈ aaaa adalah 1x0.5x0.5x0.2x0.25x0.25x0.25x0.25= 0.0009765625



Gambar III Struktur Tree dengan parsing probability 0.0009765625

sedangkan untuk struktur berikut S≈ SS ≈ NNNN ≈ aaaa memiliki peluang kemunculan 1x0.3x0.2x0.2x0.25x0.25x0.25 x0.25=0.000046875.



Gambar 3 Struktur Tree dengan parsing probability 0.000046875 .

Dengan demikian untuk mendapatkan string yang sama, kita dapat memiliki kemungkinan *parsing tree* yang berbeda. Dengan demikian, kita bisa mendapatkan *most likely parsing tree* dengan mencari peluang kemunculannya.

SCFG berfungsi untuk menentukan aturan transformasi, probabilitas suatu aturan produksi digunakan serta memetakan hasil penurunan menjadi suatu struktur.

B. Cocke Younger Kasami (CYK) Algorithm

Algoritma CYK merupakan algoritma yang diciptakan oleh Cocke Younger dan Kasami[18]. CYK merupakan suatu algoritma yang digunakan untuk parsing struktur dari sebuah sebuah *sequence*. Apabila kita memiliki suatu *sequence* berupa accgagcg dengan *grammar* sepert berikut:

N->a (0.35)	N->u (0.35)
N->c (0.10)	N->g (0.10)
S->SN (0.5)	S->NN (0.25)

Gambar 4 Contoh Grammar yang digunakan dalam CYK

Algoritma CYK menghasilkan suatu matriks 3 dimensi dimana dimensi pertama dan kedua merupakan panjang *sequence* dan dimensi ketiga merupakan jumlah variabel terminal. Untuk *sequence* diatas didapatlah matriks sebagai berikut :

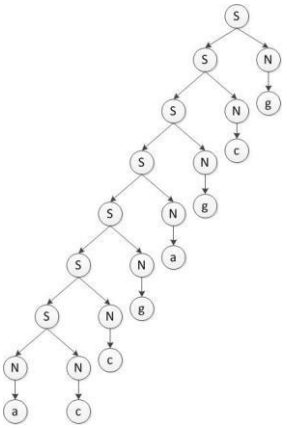
Tabel 1 Plane 0 pada matriks 3 dimensi

Plane 0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
j_1	0	0.061	0.0030	1.53 E-04	1.53 E-05	2.68 E-06	2.68 E-07	1.34 E-08
j_2	0	25	62	2.50 E-05	4.38 E-06	4.38 E-07	2.19 E-08	0
j_3	0	0.005	2.50E-04	8.75 E-05	8.75 E-06	4.38 E-07	0	0
j_4	0	0.02	0.0035	3.50 E-04	1.75 E-05	0	0	0
j_5	0	0.061 25	0.0061 25	3.06 E-04	0	0	0	0
j_6	0	0.02	0.001	0	0	0	0	0
j_7	0	0.005	0	0	0	0	0	0
j_8	0	0	0					

Tabel 2 Plane 1 pada matriks 3 dimensi

Plane 1	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
j_1	0.35	0	0	0	0	0	0	0
j_2	0.1	0	0	0	0	0	0	0
j_3	0.1	0	0	0	0	0	0	0
j_4	0.2	0	0	0	0	0	0	0
j_5	0.35	0	0	0	0	0	0	0
j_6	0.2	0	0	0	0	0	0	0
j_7	0.1	0	0	0	0	0	0	0
j_8	0.2	0	0	0	0	0	0	0

Tabel-tabel tersebut akan di terjemahkan menjadi *parsing tree* sperti berikut ini :



Gambar 5 Parse Tree sequence accgagcg

C. Grammatical Evolution

Conor Ryan, Michael O'Neill dan JJ Collins[16]. Algoritma ini digunakan untuk membangun suatu *rule* atau

fungsi. GE merupakan salah satu Algoritma Evolusi sehingga prosesnya pun mirip dengan Algoritma Genetika. GE menggunakan representasi integer dan representasi biner untuk merepresentasikan individunya. Ketika menggunakan representasi biner, setiap 8 bit kode biner tersebut diterjemahkan menjadi suatu bilangan integer dalam rentang 0 – 128 sehingga representasi integer lebih sering digunakan dalam GE. Salah satu keunggulan GE adalah penggunaan *Backus Naur Form (BNF)* untuk menginterpretasikan individu menjadi sebuah aturan atau fungsi tertentu.

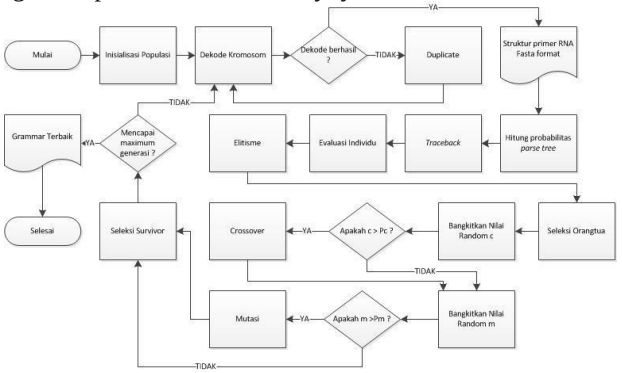
Backus Naur Form (BNF) merupakan penotasian dari sebuah *grammar*, dimana *grammar* tersebut terdiri dari *tuple* {N, T, S, P}. N merupakan himpunan simbol non terminal, T merupakan himpunan simbol terminal, S merupakan *start symbol* dan P merupakan *production rules* yang memetakan elemen N menjadi elemen elemen T.

IV. PEMBAHASAN

A. Implementasi

Proses ini dilakukan untuk mendapatkan serangkaian probabilitas *grammar* yang beragam. Secara garis besar proses ini mendekodekan *array* berisi integer menjadi sebuah *grammar*. Kemudian, *grammar-grammar* tersebut dihitung

frekuensi kemunculannya dan dievaluasi berdasarkan nilai *sensitivity*nya dimana suatu *grammar* memiliki nilai *fitness* yang baik apabila nilai *sensitivity*nya semakin mendekati satu.



Gambar 6 Flowchart Grammatical Evolution

Dimana perhitungan sensitivity dan specificity adalah sebagai berikut :

B. Skenario Pengujian

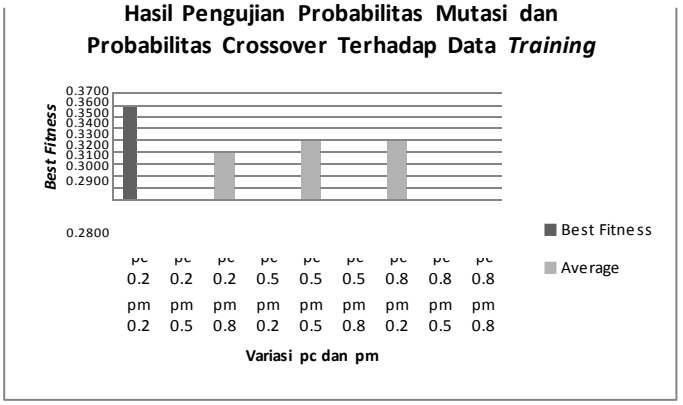
Pengujian pada penelitian dibagi kedalam tiga skenario utama yaitu pengujian probabilitas mutasi dan *crossover* , pengujian jumlah generasi dan ukuran populasi serta pengujian probabilitas *grammar* yang dihasilkan oleh GE.

V. HASIL PENGUJIAN

Berikut merupakan hasil dari ketiga pengujian dalam penelitian ini:

A. Analisis Hasil Pengujian Probabilitas Crossover dan Probabilitas Mutasi

Subbab ini menyajikan sejumlah grafik untuk memaparkan analisis pengaruh probabilitas mutasi (pm) dan probabilitas crossover (pc), seperti yang sudah dijelaskan dalam bab sebelumnya, terdapat sembilan skema pengujian probabilitas mutasi dan probabilitas crossover yang terdiri dari tiga kombinasi pobabilitas tinggi (0.8) , probabilitas sedang (0.5) dan probabilitas rendah (0.2). Berikut merupakan hasil pengukuran pengaruh probabilitas mutasi dan probabilitas crossover:



Gambar 6 Hasil Pengujian Probabilitas Mutasi dan Probabilitas Crossover

Berdasarkan hasil pengujian, dilihat dari rata-rata nilai *fitness* dan *best fitness*-nya dapat disimpulkan bahwa kombinasi probabilitas *crossover* 0.8 dan probabilitas mutasi0.2 menghasilkan probabilitas untuk setiap *production rules* dalam *grammar* Watson Crick yang membuat *grammar* memiliki nilai *fitness* terbaik.

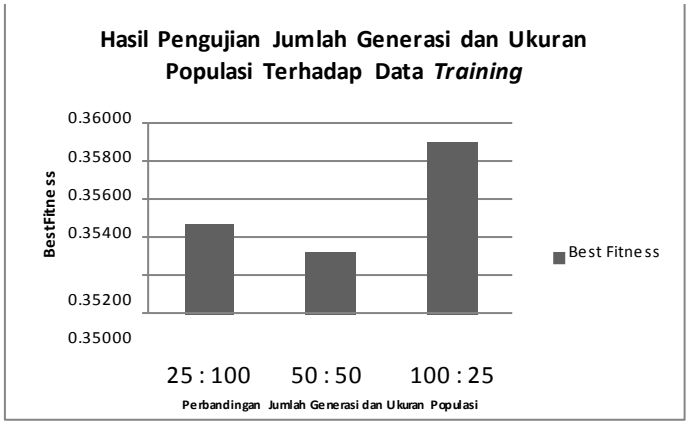
Dalam kondisi tersebut, proses *crossover* sangat sering terjadi sedangkan proses mutasinya jarang terjadi. Dengan demikian individu-individu yang berada dalam populasi merupakan individu-individu yang merupakan anak dari orangtua yang unggul dari generasi sebelumnya.

Oleh karena itu, pc 0.8 dan pm 0.2 dipilih untuk digunakan dalam skenario berikutnya karena mampu menghasilkan individu dengan nilai *fitness* terbaik dibandingkan dengan kondisi lainnya.

B. Analisis Hasil Pengujian Jumlah Generasi dan Ukuran Populasi

Subbab ini menyajikan sejumlah grafik untuk memaparkan analisis pengaruh ukuran populasi dan jumlah generasi, seperti yang sudah dijelaskan dalam bab sebelumnya, terdapat dua skema pengujian yaitu ketika populasi berukuran 50 dan memiliki generasi sejumlah 50 juga serta poulasi berukuran 25 dan memiliki generasi sejumlah

100. Berikut merupakan hasil pengukuran populasi berukuran 25 dan generasi sejumlah 100:



Gambar 7 Hasil Pengujian Jumlah Generasi dan Ukuran Populasi

Dari hasil pengujian, ketika perbandingan jumlah generasi dan ukuran populasinya 25: 100 didapat nilai *fitness* sebesar 0.3546 sedangkan ketika perbandingannya menjadi 50:50 didapat nilai *fitness* sebesar 0.3532 sedangkan ketika perbandingannya menjadi 100:25 didapat nilai *fitness* sebesar 0.3594.

Dari ketiga pengukuran terlihat bahwa nilai *fitness* tertinggi yaitu 0.3594. *Best Fitness* didapat ketika jumlah generasinya lebih banyak dan ukuran populasi yang lebih sedikit. Sehingga disimpulkan bahwa peningkatan nilai *fitness* dapat dilakukan dengan menambah jumlah generasi dalam proses *Grammatical Evolution*.

Dari skenario ini didapat *grammar* terbaik seperti dibawah ini :

Tabel 1 Best Grammar 1

S->SS:0.1	S->c:0.1
S->CH:0.1	S->g:0.1
S->GD:0.2	S->a:0.1
S->AV:0.1	S->u:0.1
S->UB:0.1	C->c:1.0
H->SG:1.0	G->g:1.0
D->SC:1.0	A->a:1.0
V->SU:1.0	U->u:1.0
B->SA:1.0	

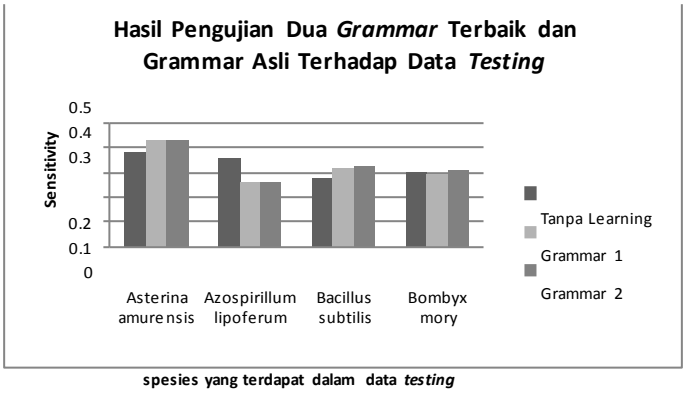
Tabel 2 Best Grammar 2

S->SS:0.16	S->c:0.083
S->CH:0.083	S->g:0.083
S->GD:0.25	S->a:0.083
S->AV:0.083	S->u:0.083
S->UB:0.083	C->c:1.0
H->SG:1.0	G->g:1.0
D->SC:1.0	A->a:1.0
V->SU:1.0	U->u:1.0
B->SA:1.0	

Kedua *grammar* tersebut akan digunakan dalam skenario berikutnya untuk diukur dan dianalisis performansinya.

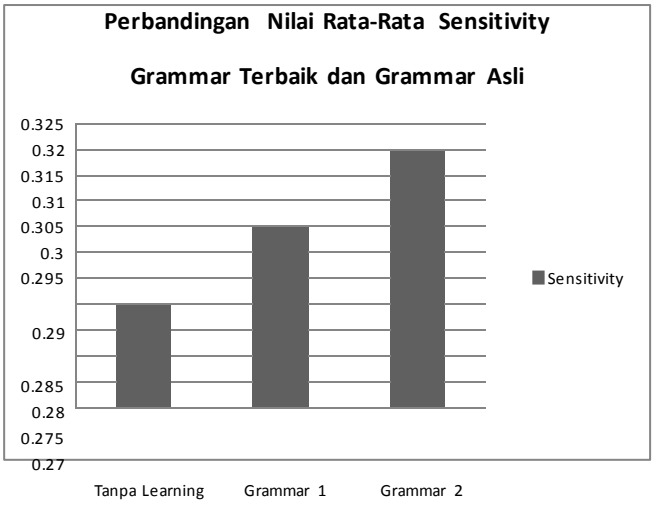
C. Analisis Hasil Pengujian Grammar Terbaik

Subbab ini menyajikan sejumlah grafik untuk memaparkan analisis performansi *grammar* dibandingkan dengan *grammar* yang tidak melalui tahapan *learning*. Pengujian dilakukan terhadap data *testing* yang terdiri dari lima spesies yaitu: Asterina amurensis, Azospirillum lipoferum, Bacillus subtilis, Bombyx Mory dan Caenorhabditis elegans. Berikut merupakan hasil pengukuran *sensitivity grammar* tanpa *learning* dan dua *grammar* terbaik hasil skenario sebelumnya:



Gambar 8 Hasil Pengujian Jumlah Generasi dan Ukuran Populasi

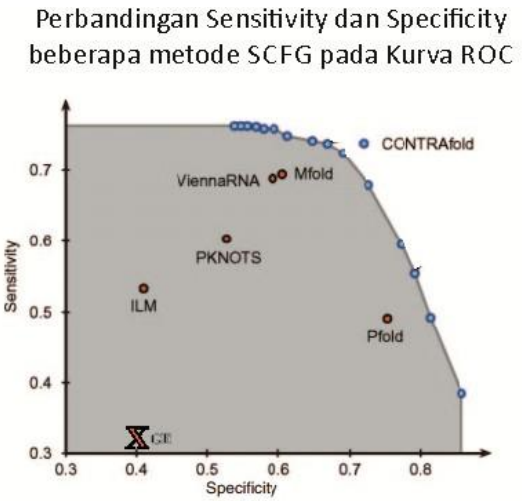
Dari grafik diatas terlihat bahwa *grammar- grammar* hasil GE seringkali memiliki nilai *sensitivity* yang lebih baik daripada *grammar* tanpa *learning*. Meskipun pada spesies Azospirillum lipoferum, *grammar* tanpa *learning* lebih baik daripada *grammar* hasil GE, tapi secara keseluruhan *grammar-gramar* terbaik hasil GE memiliki nilai *sensitivity* yang lebih baik daripada nilai *sensitivity grammar* yang tidak melalui proses *learning*. Hal ini akan terlihat lebih jelas pada Gambar 9 Perbandingan Nilai Rata-Rata Sensitivity Grammar Terbaik dan Grammar yang Tidak Melalui Proses Learning berikut ini :



Gambar 9 Perbandingan Nilai Rata-Rata Sensitivity Grammar Terbaik dan Grammar yang Tidak Melalui Proses Learning

Dengan demikian dapat disimpulkan bahwa proses GE berhasil untuk mendapatkan probabilitas untuk setiap *production rule* dari *grammar* Watson, dimana probabilitas tersebut merupakan probabilitas yang membuat *grammar* Watson Crick memiliki *sensitivity* yang cukup baik. Hal ini dikarenakan *grammar* yang dihasilkan GE memiliki nilai *sensitivity* yang secara keseluruhan lebih baik dibandingkan dengan *grammar* Watson Crick yang memiliki distribusi probabilitas secara *uniform*.

Namun untuk menilai performansi sistem ini dengan sistem prediksi struktur sekunder RNA lain yang berbasis SCFG juga, berikut ini disajikannlah kurva ROC untuk membandingkan performansi sistem pada tugas akhir ini dengan sistem lainnya :



Gambar 10 ROC Curve

Kurva ROC diatas menggambarkan performansi dari sistem-sistem lain yang berbasis SCFG. Dari kurva terlihat bahwa sistem tugas akhir ini (direpresentasikan oleh X) masih kurang baik dikarenakan terletak di daerah dibawah kurva sedangkan banyak sistem yang performansinya mendekati kurva. Meskipun telah berhasil meningkatkan nilai *sensitivity grammar* Watson Crick namun tugas akhir ini masih belum bisa dikatakan baik untuk diimplementasikan di dunia nyata.

VI. SIMPULAN DAN SARAN

Berdasarkan hasil pengujian dapat disimpulkan bahwa probabilitas mutasi dan *cross*over yang ideal untuk kasus ini adalah 0.8 dan 0.2. Probabilitas *crossover* dan probabilitas mutasi tersebut menghasilkan individu dengan nilai *fitness* terbaik yaitu sebesar 0.3594.Individu terbaik dihasilkan juga oleh jumlah generasi terbanyak dan ukuran populasi yang kecil yaitu 100 dan 25. *Grammar* yang didapat dari GE terbukti lebih baik dari *grammar* Watson Crick yang probabilitasnya terdistribusi secara *uniform*. Tujuan dari penelitian ini, untuk mengimplementasikan *Grammatical Evolution* untuk menghasilkan probabilitas dari setiap *production rules* dalam *grammar* Watson Crick telah tercapai. Namun, nilai-nilai probabilitas tersebut belum mampu mengoptimalkan *grammar* Watson Crick untuk digunakan dalam prediksi struktur sekunder RNA. Mengingat masih banyak sistem yang mendekati kurva ROC.

Dari hasil evaluasi dari penelitian ini, terdapat beberapa saran untuk mengembangkan penelitian sini pertama : mengubah skema evaluasi individu yang mempertimbangkan nilai *specificity*-nya juga. Dengan demikian, diharapkan sistem ini dapat menghasilkan *grammar* yang memiliki *sensitivity* dan *specificity* yang sama-sama mendekati nilai 1. Kedua, disarankan juga untuk mengembangkan penelitian ini dengan melakukan *learning* atau *training* berdasarkan informasi statistik strukturnya. Hal ini diharapkan dapat meningkatkan nilai *sensitivity* sekaligus *specificity* dari prediksi struktur sekunder RNA.Selain itu, disarankan juga untuk mengembangkan metode lainnya diluar SCFG untuk membangun sistem prediksi struktur sekunder RNA.

REFERENSI

[1] Anderson, James WJ, et al. "Evolving stochastic context--free grammars for RNA secondary structure prediction." BMC bioinformatics 13.1 (2012): 78.

[2] Bidargaddi, Niranjan P., Madhu Chetty, and Joarder Kamruzzaman. "Combining segmental semi-Markov models with neural networks for protein secondary structure prediction." Neurocomputing 72.16 (2009): 3943-3950.

[3] Bockenhauer, Hans-Joachim. *Algoritmhic Aspect of Bioinformatics*. Berlin: Springer, 2007.

[4] Do, Chuong B., Daniel A. Woods, and Serafim Batzoglou. "CONTRAfold: RNA secondary structure prediction without physics-based models." Bioinformatics22.14 (2006): e90-e98.

[5] Kaur, Harpreet, and G. P. S. Raghava. "A neural network method for prediction of β-turn types in proteins using evolutionary information." Bioinformatics 20.16 (2004): 2751-2758

[6] Knudsen, Bjarne, and Jotun Hein. "Pfold: RNA secondary structure prediction using stochastic context-free grammars." Nucleic acids research 31.13 (2003): 3423-3428.

[7] Knudsen, Bjarne, and Jotun Hein. "RNA secondary structure prediction using stochastic context-free grammars and evolutionary history." Bioinformatics 15.6 (1999): 446-454.

[8] Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." (2001).

[9] Liu, Yan, et al. "Segmentation conditional random fields (SCRFs): A new approach for protein fold recognition." Research in Computational Molecular Biology. Springer Berlin Heidelberg, 2005.

[10] Liu, Yan, Eric P. Xing, and Jaime Carbonell. "Predicting protein folds with structural repeats using a chain graph model." Proceedings of the 22nd international conference on Machine learning. ACM, 2005.

[11] Maaten, Laurens, Max Welling, and Lawrence K. Saul. "Hidden-unit conditional random fields." International Conference on Artificial Intelligence and Statistics. 2011.

[12] Malekpour, Seyed Amir, et al. "Protein secondary structure prediction using three neural networks and a segmental semi Markov model." Mathematical biosciences 217.2 (2009): 145-150.

[13] Ryan, Conor, J. J. Collins, and Michael O. Neill. "Grammatical evolution: Evolving programs for an arbitrary language." *Genetic Programming*. Springer Berlin Heidelberg, 1998. 83-96.

[14] Sakakibara, Yasubumi, et al. "Stochastic context-free grammars for tRNA modeling." *Nucleic acids research* 22.23 (1994): 5112-5120.

[15] Schmidler, Scott C., Jun S. Liu, and Douglas L. Brutlag. "Bayesian segmentation of protein secondary structure." Journal of computational biology7.1-2 (2000): 233-248.

[16] Suyanto, S.T., M.Sc. *Evolutionary Computation*.Bandung:Informatika,2008.

[17] Tsoulos, Ioannis G., and Isaac E. Lagaris. "Grammar inference with grammatical evolution." (2006).

[18] *Stochastic context-free grammars and RNA secondary structure prediction*. 2005. PhD Thesis. Aarhus Universitet, Datalogisk Institut.

[19] Knudsen, Bjarne, and Jotun Hein. "RNA secondary structure prediction using stochastic context-free grammars and evolutionary history." *Bioinformatics*15.6 (1999): 446-454.