A study of double assembly method for DNA sequences

Ayako Ohshiro[1,a)] Takeo OKAZAKI[2,b)] Hitoshi AFUSO[1] Morikazu NAKAMURA[2]

Abstract: To derive restored sequence without reference, some assembly algorithms have been proposed. Assembly result depends on assembly algorithm and argument. Because of that, it is difficult to obtain the best result. In this paper, we proposed a double assembly method that combine the results of different *k*-mers and different assembly algorithms in order to learn about appropriate combining to generate accurate longer sequence. We evaluated it's performance with the length, coverage ratio(cov-R), correct ratio(Cor-R) of contigs and identified effective characteristic parameters to generate accurate longer sequence.

Keywords: DNA assembly, *k*-mer, Hybrid assembly , Machine learning

1. Introduction

Thanks to development of giga sequencer platforms with parallel processing, we can derive massive short read sequences from sequencer today. Thereby that, assembly methods by use of graph that node is represented by reads such as SSAKE[1], Newbler Edena[2]. However, there are some issues in the assembly process. One of them is read-error region contained in the reads by sequencer. It makes assembly difficult, and causes misassemble. To solve this problem, assembly algorithms by use of *k*-mer have been proposed, namely Velvet[3], ABySS[4], SOAP[5].

Because there are also several path search methods for each graph, Zhenyu[6], Jason[7], Lin[8], and Steven[9] carried out comparative studies about assembly algorithms focusing structure of graph, nodes, path search algorithms. Furthermore, merging methods called hybrid assembly are developed for small genomes such as GAA algorithm[10], MAIA[11] and so on. And, assembly process increasing *k* value such as IDBA[12] and removing reads containing read-error by use of *k*-mer and constructing substring graph such as SGA[13] are also proposed.

In this paper, we proposed a double assembly method that combine the results of different *k*-mers and different traditional assembly methods by taking advantage of each other.

2. Function and problem of *k*-mer

Generally, we apply *k*-mer for read-error in sequence assembly. *k* value is any positive integer and lower than length of read. We can obtain *k*-mer as length *k* subsequences by shifting the index of position on length *l* read sequences. Hash table is generated with *k*-mer and coverage value that means *k*-mer's frequency value of hash in all read sequences. The hashes whose coverage value is especially low are deleted as read-error by sequencer. Then, we can generate adjacency graph using overlap information among *k*-mers, called de Bruijn graph. The graph has nodes represented by every *k*-mer and directed edges for every pair of *k*-mers that overlap by (*k*-1) bases. After deciding optimal paths from de Bruijn graph by path searching, we can obtain the sequence constructed by *k*-mer called contig.

Table 1,2 show the features of assembly results by changing *k* value from 15 to 27 and assemble method by use of Velvet and ABySS. We used Herpesvirus genome packaged SSAKE as experimental datasets. We represented correct, or incorrect contigs by mapping to original sequence, the number of output contigs and Max means the maximum length of them and the ratio of mapped contigs for reference by cov-R.

We can find that the difference of *k* value and assemble method have large effect on length and number of correct, or incorrect contigs. In other words, assemble result depends on specific *k* value and assemble method. Restriction of the binding condition (*k*-1 base) is considered to be one of the reason. Furthermore, there are also issues that contigs contain many incorrect contigs and they need to be eliminated before output.

Fig 1,2 show the features of mapping region on reference by vertical axis with mapping number of times and horizontal axis with reference.

We found that difference of *k* value and assemble method have large effect on mapping region. The result with large *k* value such as *k*=25-27 have small length and similar distribution. It is expected that we can generate longer contig by combining complementary distribution, for example, the result of *k*=15 (Velvet) and *k*=18 (ABySS), and the result of *k*=15 (Velvet) and *k*=25 (ABySS).

3. Double Assembly method with Heuristic approach:(DAwH)

In order to derive longer contigs, we had double assembly by combining contigs from different *k*-mers and different assembly

[1] Interdisciplinary Intelligent Systems Engineering Course, Graduate School of Engineering and Science, University of the Ryukyus, Okinawa 903-0213, Japan
[2] Department of Information Engineering, Faculty of Engineering, University of the Ryukyus, Okinawa 903-0213, Japan
[a)] ayami@ms.ie.u-ryukyu.ac.jp
[b)] okazaki@ie.u-ryukyu.ac.jp

Data	*k*	15		17		19		21		23		25		27	
30base 100000reads	contig	c	w	c	w	c	w	c	w	c	w	c	w	c	w
	Number	7	9	2	5	1	1	1	1	1	8	29	51	177	245
	Max	17176	5123	8660	12180	1196	58807	1198	58809	1200	14202	2831	3922	422	385
	cov-R	0.6		0.16		0.01		0.01		0.01		0.39		0.37	

Table 1 Assemble effection by *k*-value for each data(VELVET)

Data	*k*	15		16		17		18		19		20		21	
30base 100000reads	contig	c	w	c	w	c	w	c	w	c	w	c	w	c	w
	Number	28	24	15	11	8	9	8	4	1	4	3	1	2	1
	Max	6123	6766	11150	17178	7179	17180	11785	17182	25	28825	1195	58806	1197	58806
	cov-R	0.55		0.42		0.34		0.41		0.0004		0.02		0.02	
	k	22		23		24		25		26		27		21	
	contig	c	w	c	w	c	w	c	w	c	w	c	w	c	w
	Number	3	1	3	1	6	5	17	12	38	45	120	126		
	Max	1197	58808	3694	55131	10031	14194	6324	3331	3292	3920	1508	873		
	cov-R	0.02		0.08		0.48		0.68		0.49		0.52			

Table 2 Assemble effection by *k*-value for each data(ABySS)

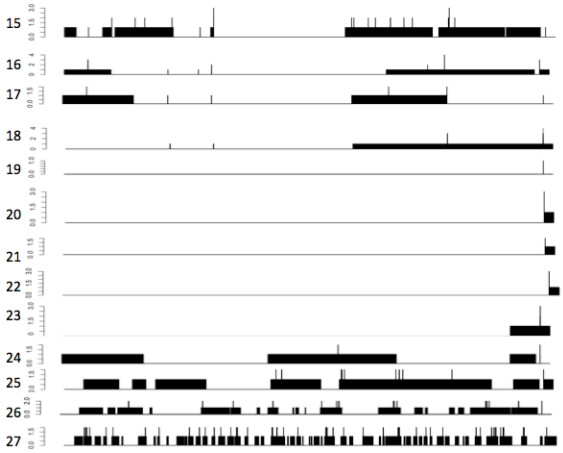


Fig. 1 Feature of mapping region by difference of *k* value : ABySS

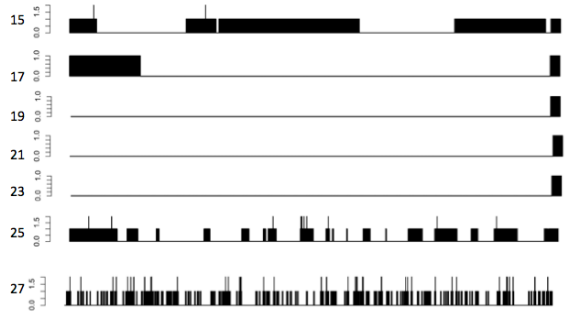


Fig. 2 Feature of mapping region by difference of *k* value : Velvet

methods(DAwH). In order to keep contig accuracy, we generated contigs based on overlap region with lower limit as follows.

(1) Prepare the whole sequence and read dataset whose base allocation is known.
(2) Obtain contigs from traditional assembly methods for any *k*-mers.
(3) Determine the combination of the *k* value and assembly method by mapping them to reference as Fig 1,2.
(4) Generate adjacency graph by all pairs of contigs with longer than 5 base overlap region.
(5) Decide optimal path with high value of overlap region.

To confirm the effect of double assembly, we carried out experiments and evaluated contig length, coverage ratio, correct ratio as Table 3.

Abbr	parameter
cov-R	Coverage rate for reference sequence
Cor-R	Correct ratio of output contigs
Longest	Longest correct contig length

Table 3 Evaluation values

We determine the combination of *k* value and assembly method as shown in Table 4 by the feature of mapping region based on Fig 1,2.

Table 4 Combinations of *k* values and assembly method

ABySS(*k*=17)+ABySS(*k*=18)
ABySS(*k*=25)+Velvet(*k*=15)
ABySS(*k*=18)+Velvet(*k*=15)

Table 5 shows the compared result of traditional and DAwH.

We could generate longer contig and increase cov-R by combining the result of different *k*-mers and assembly method compared with traditional, however Cor-R was decreased. In other words, incorrect contigs were generated by proposed method. We need more evaluation indexes to improve Cor-R by avoiding occurrence of incorrect contigs.

4. Double Assembly method with Machine Learning:(DAwML)

If we can discriminate correct combining and incorrect combining in advance, we can escape the occurrence of incorrect contigs output and improve accuracy of the assembly result. Jeong-

Method	Traditional		DAwH
k value	ABySS(k=17)	ABySS(k=18)	ABySS(k=17)+ABySS(k=18)
cov-R	0.34	0.41	0.55
Cor-R	0.47	0.66	0.29
longest	7179	11785	23427
k value	ABySS(k=25)	Velvet(k=15)	ABySS(k=25)+Velvet(k=15)
cov-R	0.68	0.6	0.91
Cor-R	0.58	0.43	0.5
longest	6324	17176	22469
k value	ABySS(k=18)	Velvet(k=15)	ABySS(k=18)+Velvet(k=15)
cov-R	0.41	0.6	0.8
Cor-R	0.66	0.43	0.3
longest	11785	17176	28820

Table 5 Compared result of traditional and DAwH

Hyeon[14] applied machine learning and predicted short reads and contigs containing read-error by use of p-value of k-mer's coverage value for each contig.

Generally, contig is composed by k-mers. Additionally, each k-mers has coverage value shown as Fig 3 and we can get $(l-k+1)$ number of k-mers from a l length contig.

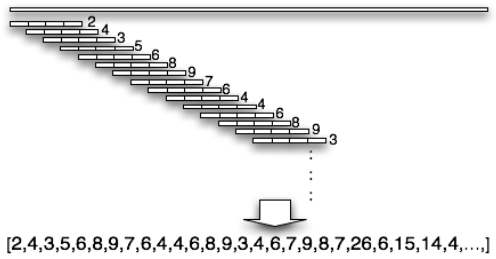


Fig. 3 Contig is constructed by k-mer

For the parameters that have relationship with the discriminant, we had each paired contig's k-mer's coverage value and distribution of it. For deciding discriminant rule from these parameters, we focused on machine learning technique, especially on the supervised learning. In order to obtain training data, we generated artificial contig data by use of all pairs of contig with overlap region more than 5 bases and determined them consistent or inconsistent by comparing to the original sequence. Finally, we obtained the discriminant rules by applying training data to machine learning. Actual procedure for getting discriminant rule is as follows.

(1) Prepare the whole sequence and read dataset whose base allocation is known.
(2) Obtain contigs from traditional assembly methods for some k-mers.
(3) Determine the combination of the k value and assembly method by mapping them to reference as Fig 1,2
(4) Generate contigs by all pairs of contigs with longer than 5 base overlap region.
(5) Distinguish restored sequences consistent or inconsistent by comparing to original sequence.
(6) Obtain the parameters
(7) Derive discriminant rules from C4.5 with the training data that consists of above parameters value and consistency judgement.

Coverage value depends on read coverage and k value. When the k value is larger, k-mer's coverage value decrease overall. We can apply p-value of k-mer's coverage value to deal them evenly. In addition, we defined the weight of overlap region about a pair of former and latter contigs by use of each contig's base coverage($basecov$) and overlap length l_{ovlp} with geometric mean as follows.

$$basecov = \frac{\sum_{i=1}^{l-k+1} \texttt{coverage value}}{l_{contig}} \tag{1}$$

$$\texttt{ovlp\_cov} = l_{ovlp} \times \sqrt{basecov_{form} \times basecov_{lat}} \tag{2}$$

As for the parameters that have relationship with discriminant, we had contig's distribution value of k-mer's coverage and base coverage for each former and latter contig as shown in Table 6. We used C4.5[15][16] as decision tree to derive discriminant rules. Decision tree construct effective rules from training data by weighting for accuracy of classifiers and output useful characteristic parameter for discriminant rule.

Abbr	Charactristic parameter
ovlp_cov	p value of overlap length of all overlap region in graph
ovlp_weight	(p value of overlap length) × ovlp_cov
mean_cov(form and lat)	mean of coverage
m_cov(form and lat)	minimum of coverage
25_cov(form and lat)	25 quartile p value point of coverage
50_cov(form and lat)	50 quartile p value point of coverage
75_cov(form and lat)	75 quartile p value point of coverage
M_cov(form and lat)	Max of coverage
SD_cov(form and lat)	standard deviation of coverage

Table 6 Characteristic parameters

We prepared the Herpesvirus sequence packaged SSAKE and generated combined sequence with combination as shown in Table 4. With the consequent discriminant rule from C4.5 for training data, we applied to the training data itself and observed learning ability. Table 7 shows discriminant result form and we defined learning accuracy as follows.

$$Le-R = 1 - \frac{num2 + num3}{num1 + num2 + num3 + num4} \tag{3}$$

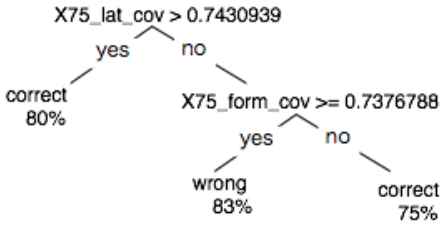
Table 8 shows the discriminant result of each double assembly combination and learning accuracy. Fig 4 shows a decision tree by the combination of ABySS(k =18)+ABySS(k=15).

ABySS(k=17)+ABySS(k=18)			
	correct	incorrect	Le-R
consistent	7	1	0.975
inconsistent	0	32	
ABySS(k=25)+Velvet(k=15)			
	correct	incorrect	Le-R
consistent	15	1	0.875
inconsistent	3	13	
ABySS(k=18)+ABySS(k=15)			
	correct	incorrect	Le-R
consistent	5	4	0.878
inconsistent	0	24	

Table 8 Discriminant results of each combination's training data

	correct	incorrect
consistent	num1 contigs judged as correct for "consistent"	num2 contigs judged as correct for "inconsistent"
inconsistent	num3 contigs judged as wrong for "consistent"	num4 contigs judged as wrong for "inconsistent"

Table 7 Discriminant result form



Decision tree constructed by training data
Fig. 4 with ABySS(k =18)+ABySS(k=15)

We could find that rules by machine learning could discriminate consistent and inconsistent contigs although with a little discriminant error from Table 8. The derived number of correct rule was greater than that of incorrect rule and k-mer's coverage with upper quantile was often regarded as effective parameter.

Because we can derive rule for collect assembly and incorrect assembly, we assigned each rule to generate the result of k-mers and assembly method, about two cases as follows.

case1 Eliminate combined contigs that don't satisfy the rule of correct assembly.

case2 Eliminate the combinations that satisfy the rule of incorrect assembly.

From the above, we proposed DAwML (Double Assembly method with Machine Learning) as follows.

step1 Prepare the whole sequence and read dataset whose base allocation is known.

step2 Obtain contigs from traditional assembly methods for each specified k-mer.

step3 Determine the combination of the k value and assembly method by mapping them to original sequence.

step4 Generate contigs by all pairs of contigs with longer than 5 base overlap region.

step5 Distinguish restored contigs 'consistent' or 'inconsistent' by comparing to original sequence.

step6 Calculate characteristic parameters for all pairs of contigs with longer than 5 base overlap region.

step7 Obtain the discriminant rules those are correct and incorrect rules from C4.5 with the training data that consists of above parameters value and consistency judgement.

step8:case1 Eliminate combined contigs that don't satisfy correct rule and output contigs.

step8:case2 Eliminate the combinations that satisfy incorrect rule and output contigs.

5. Comparative study

To confirm the effect of our proposed method, we carried out comparative experiments of traditional assembly method with specified k value , double assembly with heuristic approach (DAwH) and double assembly with machine learning(DAwML). As described in the previous section, since we could obtain correct and incorrect rule for each training data, we compared the performance for each case by evaluation value as shown in Table 3. We prepared experimental datasets as follows.

Reference length	60000
read length	30base
number of reads	150000reads
packaged	SSAKE

Table 9 Experimental datasets

Method	Traditional		DAwH	DAwML	
k value	17	18	17,18	*case*1	*case*2
cov-R	0.34	0.41	0.55	0.53	0.2
Cor-R	0.7	0.6	0.2	0.5	0.5
longest	7179	11785	23427	23427	23427

Comparative result of Traditional, DAwH and DAwML
Table 10 in the case of ABySS(k=17)+ABySS(k=18)

Method	Traditional		DAwH	DAwML	
k value	15	25	15,25	*case*1	*case*2
cov-R	0.6	0.68	0.98	0.5	0.89
Cor-R	0.44	0.58	0.51	0.78	1
longest	17176	6324	22469	6752	22469

Comparative result of Traditional, DAwH and DAwML
Table 11 in the case of ABySS(k=25)+Velvet(k=15)

Method	Traditional		DAwH	DAwML	
k value	15	18	15,18	*case*1	*case*2
cov-R	0.6	0.42	0.8	0.54	0.27
Cor-R	0.43	0.57	0.3	0.13	0.6
longest	17176	11150	28820	28820	11800

Comparative result of Traditional, DAwH and DAwML
Table 12 in the case of ABySS(k=18)+Velvet(k=15)

We found that Cor-R has improved with keeping cov-R and maximum length by comparing DAwH and DAwML-case1 from Table 17, especially about Table 18, we could remove all incorrect contigs with keeping maximum length by comparing DAwH and DAwML-case2. In Table19, we could generate longer contig but lost many correct contig by comparing DAwH and DAwML-case1 and we improved Cor-R but lost longest and many correct contigs by comparing DAwH and DAwML-case2.

6. Conclusion

In order to get longer and accurate contig independent of k value and assembly method, we proposed Double Assembly method with Heuristic approach (DAwH) and Double Assembly method with Machine Learning (DAwML).

From comparative experiments, we found that we can generate longer contig by combining the result of k value and assembly method with DAwH. However, since the result of DAwH contains many incorrect contig, discriminant rule of machine learning with DAwML is used to improve the accuracy of the assembly result.

We regarded upper quantile of k-mer's coverage for contig as high usable characteristic parameter.

References

[1] Rene L. Warren , Granger G. Sutton , Steve J. M. Jones and Robert A. Holt : Assembling millions of short DNA sequences using SSAKE, Bioinformatics, vol.23 no.4, pp.500-501, (2007)

[2] Hernandez D, Francois P, Farinelli L, Osteras M, Schrenzel J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res. 2008;18:802—9.

[3] Daniel R. Zerbino and Ewan Birney : Algorithms for de novo short read assembly using de Bruijn graphs, Genome Research, vol.18, pp.821- 829, (2008)

[4] ”Jared T. Simpson, kim Wong, Shaun D. Jackman, et al”ABySS: A parallel assembler for short read sequence data, Genome Research, vol.19, pp.1117- 1123, (2009)

[5] Li R, Li Y, Kristiansen K, Wang J. SOAP: short oligonucleotide alignment program. Bioinformatics. 2008;24:713—4.

[6] Zhenyu Li, Yanxiang Chen ”Comparison of the two major classes of assembly algorithms: overlap— layout—consensus and de-bruijn-graph ” Briefings in Functional Genomics (2012) 11 (1): 25-37

[7] Jason R. Miller, Sergey Koren, and Granger Sutton : Assembly Algorithms for Next-Generation Sequencing Data , Genomics 95 pp315-327(2010)

[8] Lin Y, Li J, Shen H, Zhang L, Papasian CJ, Deng HW : Comparative studies of de novo assembly tools for next-generation sequencing technologies, Bioinformatics. 2011 Aug 1;27(15):2031-7

[9] Steven L. Salzberg1,7, Adam M. Phillippy2, Aleksey Zimin3, Daniela Puiu1, Tanja Magoc1, Sergey Koren2,4, Todd J. Treangen1, Michael C. Schatz5, Arthur L. Delcher6, Michael Roberts3, Guillaume Marais3, Mihai Pop4 and James A. Yorke:GAGE: A critical evaluation of genome assemblies and assembly algorithms ,Genome Res. 2012. 22: 557-567

[10] “ Guohui Yao. Liang Ye, Hongyu Gao, Patrick Minx, Wesley C. Warren, George M. Weinstock ” (2012) Graph accordance of next-generation sequence assemblies, Vol. 28 no. 1, pp13-16.

[11] “ Jurgen Nijkamp, Wynand Winterbach, Marcel van den Broek,Jean-Marc Daran, Marcel Reinders, and Dick de Ridder ” Integrating genome assemblies with MAIA ” Vol.26 ECCB 2010, pp433—439

[12] “ Yu Peng, Henry Leung, S.M. Yiu, Francis Y.L. Chin ”: ”IDBA - A Practical Iterative de Bruijn Graph De Novo Assembler ”, Research in Computational Molecular Biology, 14th Annual International Conference, RECOMB 2010, Lisbon, Portugal, April 25-28, 2010. Volume 6044/2010, 426-440,

[13] Jared T. Simpson and Richard Durbin : ”Efficient de novo assembly of large genomes using com- pressed data structures ” Genome Res. 2012 22: 549-556 originally published online December 7, 2011

[14] Jeong-Hyeon Choi1,*, Sun Kim1,2, Haixu Tang1,2, Justen Andrews1,3, Don G. Gilbert1,3 and John K. Colbourne1:’A machine-learning approach to combined evidence validation of genome assemblies’ Vol. 24 no. 6 , pp. 744—750,(2008)

[15] J. Ross Quinlan Morgan Kaufmann, San Mateo,CA: ”C4.5:Programs for Machine Learning”(January 1993)

[16] Thomas G. Dietterich:’Machine-Learning Research Four Current Directions’ (AAAI) AI Magazine Volume 18 Number 4 (1997)