Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 23 (2013) 122 – 128

Procedia
Computer Science

4th International Conference on Computational Systems-Biology and Bioinformatics, CSBio2013

# CalcGen sequence assembler using a spatio-temporally efficient DNA sequence search algorithm

Kyong Oh Yoon, Sung-Bae Cho[*]

*Department of Computer Science, Graduate School of Engineering, Yonsei University, Seoul 120-749, South Korea*

**Abstract**

The advent of ultra-high-throughput sequencing technology produces an enormous amount of bio-sequence information. Also, the current advances in the bio-industry bring forward the era of personalized medicine using individual genome information. However, the analysis of massive number of bio-sequences requires large storage, so that analysis sometimes needs supercomputer and novel software that can handle such volume of sequence information. For that type of analysis, several sequence match algorithms have been devised in terms of alignment and assembly, which are fundamental for analyzing bio-sequences. Those algorithms regard nucleotide sequences as strings and compare characters one-by-one during analysis of sequences. They use hash index tables, de Bruijn graph, Burrows-Wheeler transform method, and so on. In this paper, for time and space efficient DNA searching, we propose a simple algorithm that transforms base sequence into k-mer integer array and then we analyze the integer array transformed by unit search operator and non-unit search operator, resulting in a storage space reduction of about 0.28 fold. Furthermore, based on the proposed algorithm, we have developed a sequence analysis program called CalcGen assembler, and show the usefulness of the program with several experiments.

*Keywords :* next-generation sequencing (NGS); sequence search algorithm; assembly analysis

* Corresponding author. Tel.: +82-2-2123-2720;  fax: +82-2-365-2579.
*E-mail address*: sbcho@cs.yonsei.ac.kr

1877-0509 © 2013 The Authors. Published by Elsevier B.V.
Selection and peer-review under responsibility of the Program Committee of CSBio2013
doi:10.1016/j.procs.2013.10.016

## 1. Introduction

A tremendous amount of biological data has been stored these days by the advent of ultra-high-throughput sequencing technology. On 23 November 1999, the publicly funded human genome project held a massive, worldwide celebration to mark the completion of one billion base pairs (bp), one-third of the way to the full sequence of the human genome. Today, sequencing one billion bp can be done within hours in any lab equipped with an Illumina GAII or ABI SOLiD 'second generation' sequencing machine and the work of mere minutes in large-scale sequencing centers[1]. Furthermore, modern medical science is developing personalized medicine that pursues the diagnosis and therapy through individualized genetic data analysis. Yet only a few scientists are studying more advanced new nucleotide sequencers. Using those genetic data spurt by next-generation sequencer, scientists can study genetic functions and genetic regulations using new organism genetic sequences, comparative genomics, single nucleotide polymorphism (SNP), copy number variation (CNV), and others.

The most important first step in understanding next-generation sequencing data is the initial alignment or assembly that determines whether an experiment has succeeded and provides a first glimpse into the results. In those alignment programs, there are currently MAQ, SOAP, ELAND, SHRiMP, ZOOM, BFAST, MOSAIK (https://code.google.com/p/mosaik-aligner/)[2,3], and Bowtie, BWA and SOAP2 that are based on Burrow-Wheeler transformation[4,5]. In assembly programs, there are SHARCGS, VCAKE, VELVET, EULER-SR, EDENA, ABySS, and ALLPATHS that are based on de Bruijn graph data structure[6,7,8,9].

As some programs depend on next-generation sequencing machine and have limitation of memory usage, memory errors or runtime errors occur frequently during analysis processes. Therefore, some analysis programs such as SOAPdenovo run on the environments of supercomputer that have many nodes or need huge physical storage [10]. The development of nucleotide sequence analyzer leads to elongated length of analyzed sequence and produce bulky amount of sequencing data. We need algorithms that use less physical memory storage and disks, and rapid speed of analysis in the field of bioinformatics. A lot of researchers are conducting in-depth studies for the objects.

In this paper, we propose a spatio-temporally efficient DNA sequence search algorithm and analysis program that reduces the amount of disk and memory usage of sequence analysis.

## 2. Spatio-temporal efficient DNA sequence searching

The proposed spatio-temporal DNA sequence search algorithm is illustrated in Fig. 1. First, sequence data are transformed from string into integer arrays by k-mer unit for space efficiency and computational simplification. The size of transformed DNA sequence data depends on user defined k-value. Second, we perform match analysis of transformed DNA sequence data by k-mer unit match and $I_{search}$ match. In comparison process of k-mer size we use k-mer unit match. When the size of compared DNA sequence is less than k-value, we use $I_{search}$ match.

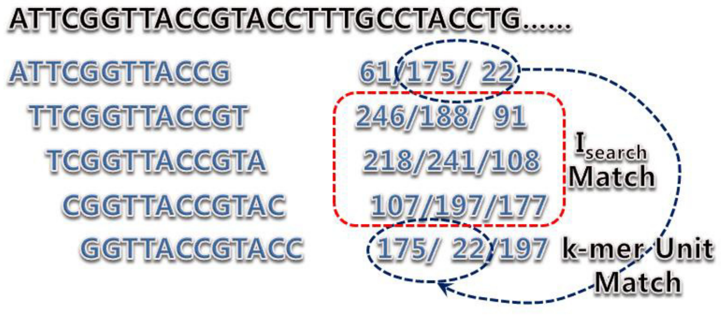
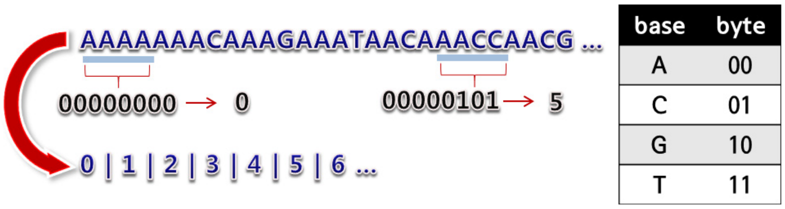


Fig. 1. Sequence search with data type transformation (k-value is 4)

## 2.1. Integer transformation of DNA sequence data

The genetic material of organisms has basic units: adenine (A), guanine (G), cytosine (C), and thymine (T). When genome samples are analyzed, we can get raw data that are full of one dimensional long string of A, C, G, and T. In this paper, we transform one nucleotide data from character into 2 bit, bind these nucleotides by k-mer unit, and transform bind unit into integer. This step is illustrated in Fig. 2(“|” is separator).



| base | byte |
|---|---|
| A | 00 |
| C | 01 |
| G | 10 |
| T | 11 |

Fig. 2. Transformation of nucleotide sequence into integer array (k-value is 4)

The efficiency of transformation of nucleotide sequence into integer array depends on k-value and integer size theoretically. The equation for evaluation is as follows.

$$E_{save} = \frac{I_{len}(S_{len}k^{-1})}{8S_{len}} = \frac{I_{len}}{8k} \tag{1}$$

In the above equation, $E_{save}$ is storage efficiency, $S_{len}$ is the length of nucleotide sequence, $I_{len}$ is bit number of integer type, and $k$ is the length of k-mer. In the case of 4-byte integer, because the size of integer is 32bit, if the length of k-mer is more than 4, storage reduction efficiency is less than 1, and the length of k-mer has never to be over 16. In this case, storage reduction efficiency $E_{save}$ is theoretically 0.25 and we can save approximately 3GB data of human genome into 750MB transformed data.

## 2.2. Computation of DNA sequence match: Unit and non-unit operators

When we transform nucleotide sequence into k-mer integer array and the data have a storage efficient data structure, we can search rapidly nucleotide match region by group match, but we cannot search it by that method in case that the start position of match is not equal to the start position of k-mer (see Fig. 1). In this case, we can search nucleotide match region by non-unit operator as shown in the following equation.

$$I_{search} = 4^{len}(RI_n \% 4^{k-len}) + (quotient)\left(\frac{RI_{n+1}}{4^{k-len}}\right) \tag{2}$$

In the above equation, $I_{search}$ is compared integer number, $k$ is the length of k-mer, *len* is difference of the start positions between the match and k-mer, $RI_n$ is the value of the current searching array that has match difference, $RI_{n+1}$ is the value of next array, % is mod operator, and (quotient) is function that takes only quotient value from number. This % function of the above equation cuts the front of difference between sequence matches, and the *quotient* function adds a part of next sequence that is the length of *len*. By these computations we can search nucleotide matches that have different start position between k-mer and match sequence. When we reach the last sequence that has less size of k-mer, we recognize that this match is valid if it satisfies the following equation.

$$4^{len}(RI_n \% 4^{k-len}) \leq I_{target} \leq 4^{len}(RI_n \% 4^{k-len}) + 4^{len} - 1 \tag{3}$$

In the above equation, $I_{target}$ is the last integer array of the compared sequence. That equation means that if the last sequence has any different bases of mismatch length, that match is valid.

## 2.3. CalcGen assembler

The fundamental analysis for DNA nucleotide sequence is alignment and assembly. Because assembly is more complex and needs heavy computation, we have implemented CalcGen assembler that includes a spatio-temporally efficient DNA sequence search algorithm we have devised.

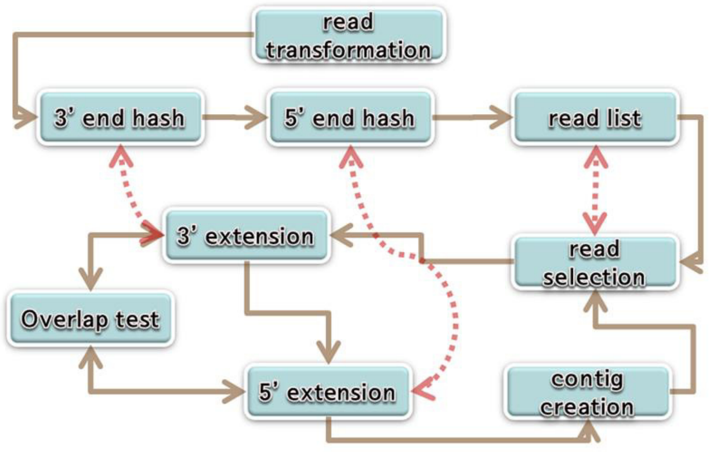


Fig. 3. CalcGen assembler process

Fig. 3 is the diagram of CalcGen assembler process. In this diagram solid arrow denotes process flow and broken array denotes data reference.

This program receives fasta format file of short read, size of k-mer, and minimum size of overlap, and then conducts assembly process. First, read transformation process is to transform read sequences produced by next generation sequencer into integer array data that have size of user-defined k-mer. Then, it creates two hash tables and one linked list used assembly process. 3' end hash is hash table for 3' extension of contig that has keys of the first array value of 5' k-mers and values of read ID. 5' end hash is hash table for 5' extension of contig that has keys of the last array value of 3' k-mers and values of read ID. The linked list is used when this program reduces analyzed read ID and selects unused read ID. This program uses another linked list for contig creation that has read IDs and the start positions of overlaps, and the start position is 0 when the compared sequences are identical.

After creation of data structures this program selects unused read and search matches between the first integer array of selected read and other reads using 3' end hash. If there are valid matches, this program analyzes whole overlap of residual sequence of compared reads using the spatio-temporally efficient DNA sequence search algorithm. If there is not valid match in hash tables, $I_{search}$ match starts with unmatched value (*len*) between 1 and k-mer value. In this two-step search process, if there is not any valid match, extension process is terminated. Otherwise, valid overlapped read sequence is found, and searches other matches between this overlapped sequence and unused other read sequences to extend current contig. In this process, read ID connected contig is removed from 3' end hash, 5' end hash, and read linked list.

After the extension of 3' extension, 5' extension of contig is executed. This program searches matches between the first selected read of 3' extension process and other unused read using 5' end hash, and tests overlap of these sequences using unit or non-unit operator like above 3' extension processes.

Through 3' and 5' extension processes one contig is created. Then, to create another contig this program selects another read sequence that is left in read linked list and the above assembly processes are conducted repetitively until there is not read ID in read linked list.

## 3. Experiment and result

We compare the proposed CalcGen program with ABySS, SHARCGS, Velvet, EDENA, and SSAKE that are frequently used in the previous studies and can set up on our computing system, HP DL 160 G5 server and Ubuntu 11.04 Linux operating system. For evaluation of efficiency, we use simulated nucleotide sequence of BAC clone size, 262,323bp, and create each 2,000,000 read sequences of sizes, 32bp, 70bp and 125bp.

We set the value of k-mer to the best efficient value that is found by searching the optimal value of k-mer as shown in Fig. 4.

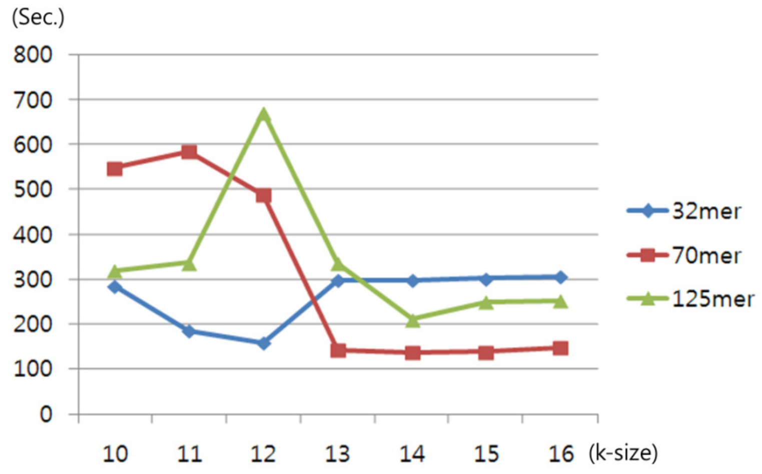


Fig. 4. Execution time of CalcGen assembler by the value of k-mer

According to the figure, in the case of 32-mer read sequence, it takes short execution time when the value of k-mer is 12. In other cases of 70-mer and 125-mer read sequences, it takes short execution time when the value of k-mer is 14.

The result of comparison between CalcGen assembler and the previous other assemblers is as shown in Fig. 5.

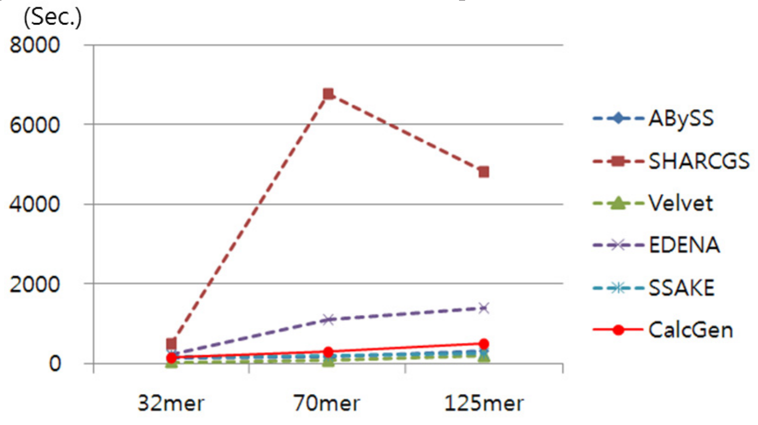


Fig. 5. The result of execution time

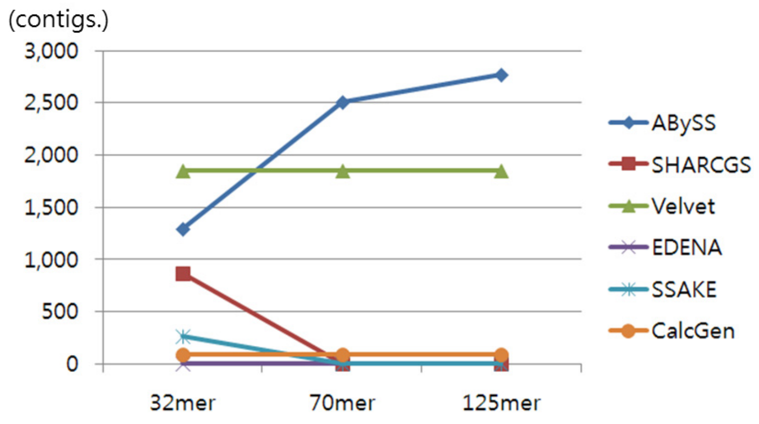


Fig. 6. The result of created contig number

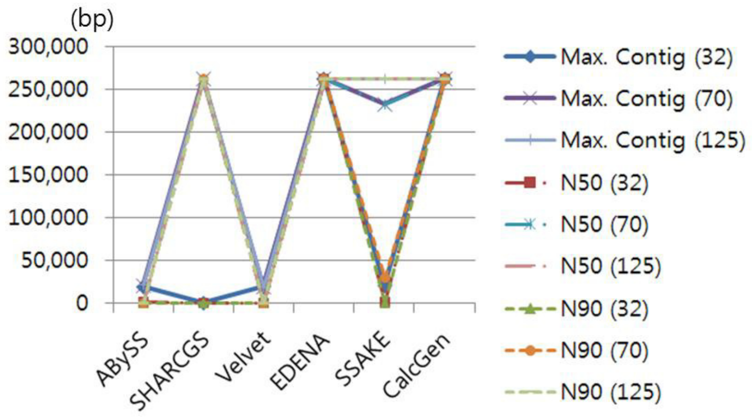


Fig. 7. The result of maximum contig size and N50, N90

According to the experiments CalcGen assembler that we propose takes shorter execution time than other known assemblers (SHARCCGS, EDENA) (see Fig. 5). But comparing between CalcGen and the two assemblers, Velvet and SSAKE, CalcGen is a little faster or slower. In the case of 32-mer read sequence, execution time of CalcGen is 157 seconds, but those of the other assemblers, ABySS, SSAKE, EDENA, and SHARCGS are 166, 194, 235, and 490 seconds, respectively, but that of Velvet is 47 seconds. This result shows that the proposed algorithm is temporally efficient. Assembly results are measured by the size and accuracy of their contigs and scaffolds. Analyzing assembly results, assembly size is usually given by statistics including maximum length, average length, combined total length, and N50. The measurement, N50 is the length of the smallest contig in the set that contains the fewest (largest) contigs whose combined length represents at least 50% of the assembly. N90 is 90%. Observing other results in figures 6 and 7, the proposed algorithm is better than the other known assemblers in the number of contigs and maximum contig size. This result shows that our algorithm has better overlap search accuracy and contig creation efficiency than the other assemblers.

To confirm the storage efficiency, we perform the experiment to transform chromosome 1 of human genome into integer array type data. The result of this experiment is as shown in Table 1. The results show that there is the difference between storage efficiency and theoretical efficiency. This difference comes from existing new line and form feed characters in source fasta file. We did not transform new line or form feed character and produce connected integer arrays.

Table 1. Storage efficiency by integer transformation.

| k-value | Source | 10 | 12 | 14 | 16 |
|---|---|---|---|---|---|
| File size | 253 MB | 100 MB | 83 MB | 71 MB | 62 MB |
| Storage efficiency | - | 0.39 | 0.33 | 0.28 | 0.25 |
| Theoretical efficiency | - | 0.40 | 0.33 | 0.29 | 0.25 |

## 4. Concluding remarks

The analysis of bulky sequencing data from next generation sequencer needs large storage, main memory space, and it takes a lot of time. In this paper, we have attempted to overcome these problems using a new approach to transforming fasta formed text file of nucleotide into integer array binary file by k-mer size, and invented unit and non-unit operators that solved the k-mer unit match problem by the specific equations. Moreover, we implemented CalcGen assembler that conducted heavy calculation of assembly process. This assembly program showed shorter execution time and better accuracy than the known assembly programs, and reduced storage size (maximum 0.25 fold).

In this paper, CalcGen program that we implemented is a fundamental algorithm using transformed integer type array data. In real genome analysis, there are many assembly problems such as sequencing error, repetitive sequences (repeats), and so on. In our further studies, we will work out those problems and implement alignment program using the proposed search algorithm.

## References

1. Flicek P, Birney E. Sense from sequence reads: methods for alignment and assembly. Nature methods supplement; 2009. vol.6 no.11s, p.S6-S12.
2. Li H, Ruan J and Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res; 2008. vol.18, p.1851-1858.
3. Li R, Li Y, Kristiansen K and Wang J. SOAP: short oligonucleotide alignment program. Bioinformatics; 2008. vol.24, p.713-714.
4. Langmead B, Trapnell C, Pop M and Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol; 2009. vol.10, R25.
5. Li H and Durbin R. Fast and accurate short read alignment with Burrows Wheeler transform. Bioinformatics; 2009. vol.25, p.1754-1760.
6. Dohm, JC, Lottaz C, Borodina T and Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res.; 2007. vol.17, p.1697-1706.
7. Zerbino DR and Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res.; 2008. vol.18, p.821-829.
8. Hernandez D, François P, Farinelli L, Osterås M and Schrenzel J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res.; 2008. vol.18, p.802-809.
9. Simpson JT et al. ABySS: a parallel assembler for short read sequence data. Genome Res.; 2009. vol.19, p.1117-1123.
10. Li R et al. De novo assembly of human genomes with massively parallel short read sequencing. Genome Res.; 2010. vol.20, p.265-272.