# Heuristically Tuned GA to Solve Genome Fragment Assembly Problem

Satoko Kikuchi, Goutam Chakraborty
Graduate school of Software and Information Science
Iwate Prefectural University
020-0193 Iwate, Japan
{g231c013@edu., goutam@}soft.iwate-pu.ac.jp

***Abstract*— We proposed a genetic algorithm (GA) approach to solve the genome sequencing problem. The main contribution of this work is to add two ideas to improve the efficiency of the algorithm - (1) a Chromosome Reduction Step (CRed) method to shorten the length of the chromosome and thereby the search-space, and (2) Chromosome Refinement Step (CRef) is a greedy heuristics to locally improve the fitness of chromosomes. The algorithm will bring out longer and longer contigs with shorter and shorter gaps, as it continues running. At any stage the user can view the result, stop it when the output serves her/his purpose, or continue for getting longer contigs. We ran the proposed algorithm on part of the *Wolbachia* project work, and compared the results.**

## I. Introduction

At present, many research groups are dedicated to analyze genomes of various living objects, and DNA sequencing of the whole genome is the primary goal. The length of genomes to investigate are ever growing, from small viruses (a few thousand nucleotides) to large mammals ($\sim$ 3 giga nucleotides), whereas a DNA sequence up to a length of only $10^3$ nucleotides could be read. The prevailing methods is to fragment the whole genome, read the ends of fragments, and then use a computer program to assemble them in proper order to infer the whole genome sequence. This is a NP-hard problem. Several deterministic algorithms based on graph-theory, and greedy heuristic algorithms are proposed. But they are computationally intensive. The main motivation of this work is to find an efficient fragmentation assembly algorithm that could run on cheap computers, yet able to find nearly correct draft sequences.

Due to exponential growth in computing power, many biological problems, which were too complex, now came within the computational limits. Due to sophisticated instruments, the biological activities of the living organisms are now getting available in bits and pieces of information and the field which tries to join those pieces of information together and explicate it, is called Bioinformatics [1] [2]. Problems like homology search, clustering of the of gene expression data, research on the 3-D structure of proteins, the genome sequence, are to name a few. Many of the problem involved in Bioinformatics are string matching or sequencing problems, which are NP-complete and require high end computers and long execution time. The main motivation of this work is to propose an efficient algorithm for *assembling DNA fragments*.

### A. Genome Preliminaries

A genome is formed by a sequence of four types of molecules, called nucleotides or bases, namely A (adenine), T (thymine), C (cytosine) and G (guanine) [3] [4]. Proteins are responsible for different functions of living organisms, and are formed by a sequence of amino acids. The 20 amino acids are again coded by triple-bases called codons, e.g., AAA is for Lysine, GAA for Glutamic [4] etc.. Thus the *codons* are words of length 3 formed by the alphabet set {A, T, C, G}. Only 20 out of $4^3 = 64$ possible codes are used, and many codes map to the same amino acid, like AAA and AAG both map to Lysine, ACA, ACG, ACT, and ACC all map to Threonine. The DNA sequence is thus responsible for producing different proteins, and therefore is at the root of functioning of a living organisms. Decoding genome sequence is thus vital to understand the function as well as malfunction of living things. The genome sequence information is vital for medical, agricultural and many other research area.

### B. The Problem of Fragment Assembly

*Gel electrophoresis* is the common laboratory method for reading a DNA sequence. It can read on an average of a mere 500 to 800 base pairs from a larger sequence. But our quest to know genome sequence is ever-increasing, both in varieties of organisms and the length of the genome. Most commonly used and cost effective process to find genome sequence is *shotgun sequencing* [5] [6] [7]. The basic principle is to first clone the target sequence into multiple copies, then break them into fragments of nearly equal lengths, read the sequences at both ends of the fragments, and finally reassemble them in proper order to recover the target genome [6] [7]. Computer science comes in picture in the last step of assembling the fragments. The shotgun sequencing approach was first introduced by Fred Sanger in 1982 [8], and was thought to be able to sequence to a maximum of 30 Kbps to 50 Kbps. In fact, during 80's it (*shotgun sequencing*) could successfully sequence up to 10 Kbps, and by 1990 it could sequence segments up to 40 Kbps.

In 1995, Fleischmann et al. [9] could assemble the ≈1,800 Kbps long *H. Influenzae* bacterium, and in 2000 Myers et al. [10] was able to assemble ≈130 Mbps long *Drosophila* genome. By 2001, Lander et al. [11] presented an initial sequencing of human genome of ≈3.5 Gbps length. This was possible, not because *electrophoresis* can now read longer base pairs, but due to new innovative algorithms to assemble fragments and improved hardware to crunch them. During last decade many assembling algorithms were proposed, the important ones being TIGR assembler [12], Consed [13], CAP3[14], ARACHNE [15], AMASS [16], EULER [17], RECOMB [18], Phrap [19], *454* [20]. A good survey of many of these algorithms is available at [7]. During last ten years a few works were reported [21] [22] [23] to use genetic algorithm [24] to solve fragment assembling problem. Our work is also based on genetic algorithm. The main contribution of this work is to add two ideas to improve the efficiency of the algorithm - (1) a Chromosome Reduction Step (CRed) to shorten the length of the chromosome and thereby the search-space, and (2) Chromosome Refinement Step (CRef) to locally improve the fitness of chromosomes by some greedy mutation. Results are compared with that obtained from *Wolbachia* genome project.

The paper is organized in the following sections. In section 2, shotgun sequencing and problems of the existing techniques are briefly explained. Section 3 is devoted to the proposed algorithm. In section 4, we explain the three experiments we did and their corresponding results. Conclusion is in section 5.

## II. Shotgun Sequencing Method and Its Present Status

### A. Shotgun Sequencing

As already mentioned, even today it is possible to read only a length of 500 to a maximum of 1000 base pairs by *electrophoresis* method. To decode a long DNA sequence we need to fragment it, read the individual fragments and then assemble. This is called *shotgun sequencing*, and is the basis of all sequencing strategies. Initially it was thought that the only way to read large genomes is to divide the whole genome into large pieces called BACs (*bacterial artificial chromosomes*), which are then mapped to the genome. Shotgun sequence is to be used to sequence each BAC. It is a two step hierarchical process.

In contrast, WGSS (*whole genome shotgun sequence*) endeavors to do the sequencing directly from the fragments, skipping the BAC step. It was thought to be computationally too heavy, and difficult due to repeat stretches in the genome. Yet, in 2000 Myers et al. successfully sequenced the fruit fly *drosophila* genome of length ≈125 Mbps using WGSS [10]. and WGS was established as a general technique.

Though the base sequence deciphered by WGSS might contain gaps, and the accuracy is lower than clone-by-clone shotgun sequencing by using BACs genome map, yet, in many genome researches, rough or partial information of base sequences might be good enough. With that in mind WGSS was used in the determination of draft human genome in 2001 by Celera Genomics [25]. In Japan too, WGSS was used to decode genome of *Silkworm* by Mita in 2004 [26], and the genome of *Aspergillus oryzae* was decoded by Machida in 2005 [27]. Our target is similar. The algorithm will bring out longer and longer contigs with shorter and shorter gaps, as it continues running. The user can view the result, stop it when the output serves her/his purpose, or continues for getting longer contigs. Moreover our algorithm gets more and more efficient with generations due to CRed and CRef operations.

### B. Outline of WGSS

The whole process of WGSS is divided into two - one is the biological part of cloning, fragmenting, and reading, and the other one is the computational part of assembling the fragments.

*1) Biological Part:* The basic shotgun procedure starts with a large number of copies of DNA whose sequence we need to find out. The genome is then physically cut into a large number of random fragments. Fragments that are too large or too small are then discarded. The length of short fragments are about 2kbp, and the long ones are about 10kbp. The fragments are then inserted into the DNA of a bacterial virus (phage), called *vector*. Typically one vector contains one fragment. The fragments are called *inserts* and the set of inserts, a *library*. Next, a bacterium is infected with a single vector, which generates clones of the vector as well as the insert (the fragment) within it. Then, the base pair at both ends of all the fragments are read with DNA sequencer as shown in Fig.1. Only about 500 to 1000 bp can be read using present sequencer technology. This read length depends on the passing speed in the capillary of the sequencer. But even done meticulously a read length of more than 1000 bp is not possible. The base sequence at both ends of fragment read by the sequencer is called *read*, and the pair of reads from two ends is called *mate-pairs*. This procedure is shown in Fig. 1.

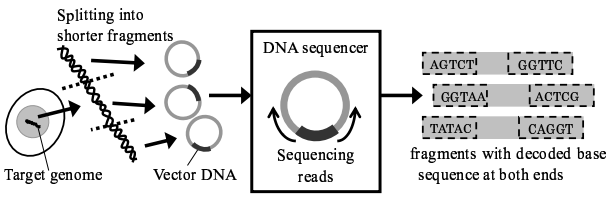


Fig. 1. Shotgun sequencing

With large number of clones of the vectors, finally the total base pair reads of fragments is several times than the number of bases of the original genome. Here, we use a term *Coverage* which is a measure of redundancy of the fragment data, and is defined as the number of bases read from fragments as a ratio of the length of the source DNA [28].

$$Coverage = \frac{\sum_{i=1}^{n} reads\_of\_fragment_i}{target\_genome\_length} \quad (1)$$

It is considered that, to be able to reconstruct the original genome, the *coverage* should be set around 8 to 10 (described as 8X$\sim$ 10X). If coverage is high, the probability of covering original genome is more and the accuracy is improved. However, the number of fragments and therefore the computational complexity also increase. In practice, to sequence large genomes, hundreds of thousands to tens of millions of fragments are used for assembly. Even then some parts of the original genome may not be reconstructed, as this is after all a stochastic process.

*2) Computational Part:* To sequence the Original DNA, we first identify overlapping sections by comparing the already read base sequences at both ends of the fragments, as shown in Fig.2. Long ranges of base sequences without gaps, obtained by assembling, are called *contigs*. Fig.2 shows two contigs formed. Here, it is presumed that two overlapping read, one a prefix of a fragment and the other the suffix, originate from the same region of the genome. This is however not always true as there could be repetitive sequences in the original genome.

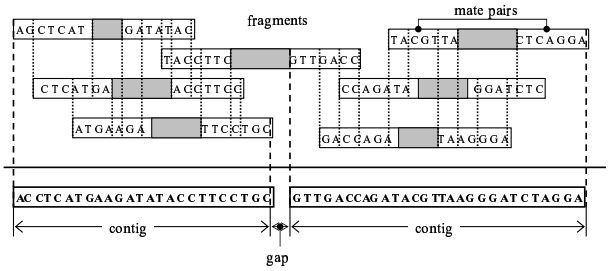


Fig. 2. Formation of contigs

The existing techniques compare all fragments for overlap detection using distributed processing with a large number of high performance computers. Celera Genomics in human genome project reported that "Computing the set of all overlaps took roughly 10,000 CPU hours with a suite of four-processor Alpha SMPs with 4 gigabytes of RAM. This took 4 to 5 days in elapsed time with 40 such machines operating in parallel" [25]. Obviously, such computational support is still too expensive.

The position and distance between contigs are determined from the mate pair of fragments (Fig.3). Subset of contigs with known order and orientation are grouped together and this process is called scaffolding. This is done by constructing a graph in which the nodes correspond to contigs, and a directed edge links two nodes when mate-pairs bridge the gap between them. Most of the recent assemblers include a scaffolding step. A rough frame of original genome sequence is made by this scaffolding process. After all contigs are oriented and ordered correctly, we can close gaps between two contigs. This process is called gap closer or finishing.

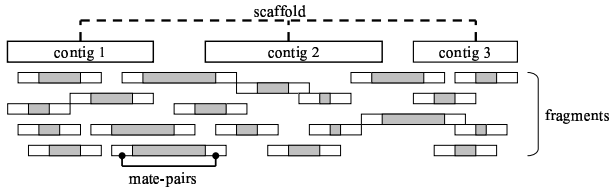


Fig. 3. Scaffolding

The finally obtained base sequence that is nearest to the original genome sequence is called consensus sequence. And the above mentioned procedure based on contig formation and scaffolding to form consensus sequence is called overlap-layout-consensus paradigm. Many of important assemblers are using this paradigm. For example, Celera assembler employs scaffolding algorithm based on graph theory using mate pairs. TIGR assembler employs greedy algorithm where two fragments with largest overlap scoring are merged together and this is repeated until no more merges can be done.

### *C. Issues with Shotgun Sequencing*

Most of the existing assemblers are owned by large-scale research facilities. They are distributed processing systems consisting of a large number of interconnected high performance computers. To deploy, or even to rent such a system is enormously costly. Though some of the assembling softwares are available to the public, in most of the cases those algorithms' user interfaces are unclear and it is hard to transport programs to one's own system. Many of the algorithms use exhaustive or computationally intensive heuristics, involving number of comparisons increasing exponentially with the number of fragments. Moreover the fragment assembly process goes through several phases and number-crunching is required at each phase.

On the other hand, the need for genome sequencing is felt more and more strongly at every small medical research centers, drug development centers, agricultural research centers etc.. To help progress their research activities we need an efficient fragment assembling algorithm, which could run on an ordinary PC. Moreover, on many occasions what one needs is only a partial sequencing, and not the sequence of the whole genome.

Genetic Algorithm (GA) is already a proven robust algorithm for graph searching and many other exponential combinatorial NP-hard problems. There are already a few researches of fragment assembly using different varieties of GA, like hybrid GA which includes heuristic algorithms, SAGA where GA is conflated with simulated annealing (SA) [29], distributed GA [22]. Our proposed GA approach is standard genetic algorithm like the one proposed by Parsons et al. [21]. However, the main contribution of this work is to add two ideas to improve the efficiency of the algorithm - (1) a Chromosome Reduction Step (CRed) to shorten the length of the chromosome and thereby the search-space, and (2) Chromosome Refinement Step (CRef) to locally improve

the fitness of chromosomes (a type of greedy algorithm). Moreover, the user can consult with the intermediate result after every few thousand generations of GA run. Depending on the quality of the results and her/his requirements, the genetic search may either be stopped or be allowed to continue to run. With further generations the efficiency of genetic search improves due to CRed and CRef steps.

## III. Proposed Technique using Genetic Algorithm

We proposed method for genome sequencing using GA. The basics of GA [30] is omitted here as it is well known. As our proposed technique is a specialized setup in fragment assembly, we emphasize the relevant aspects only.

### A. The Style of GA Chromosome

Though binary or continuous value is used as genes of GA chromosome in general, we directly used fragments as genes for fragment assembly. First, the test genome string is cloned and cut at random locations imitating the process done in WGSS. The fragments are labeled in serial numbers, 1 to $N$, and the read information are stored in a Table. Here, $N$ is the total number of fragments. The chromosome of GA, composed of all these fragments as its gene, is constructed. Thus the chromosome is actually a permutation of numbers 1 to $N$. The information of each gene, i.e., the base sequences of both ends of fragments are known.The fragments come in random sequences in different chromosomes. The base sequence of a fragment is not cut on the way of genetic operations like crossover, because crossover and mutation are done at the boundary of fragments.

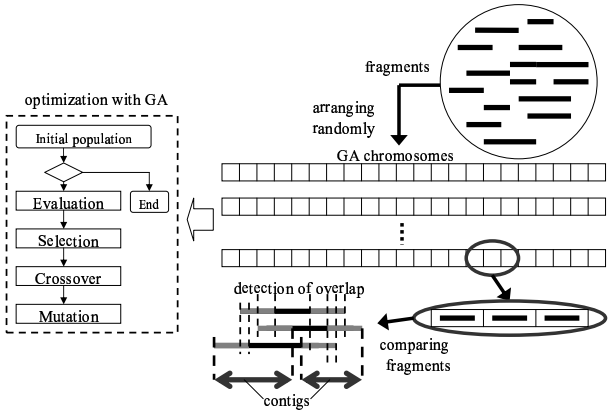


Fig. 4. Chromosomes of GA for fragments assembly

### B. Evaluation Function

For the evaluation, i.e., fitness calculation, the similarity of base pairs of the adjoining gene (actually the genome fragments) in the individual chromosomes, are calculated.

$$Fitness\,(c) = \sum_{i=0}^{n-2} similarity\,(i, i+1) \qquad (2)$$

The sum total of the similarity is the fitness of an individual as in illustrated in Fig.4. In Eq.(2), $c$ is a chromosome, $i$ and $i+1$ are adjoining fragments and $n$ is the total number of gene in the chromosome. To calculate the similarity we use Smith-Waterman algorithm that detects a local alignment by dynamic programming.

### C. Selection

Initially the fitness did not show a considerable disparity in chromosomes because of the random permutation of the fragments. Therefore we use ranking method in GA selection. We also use elitist preservation.

### D. Crossover and Mutation

If we allow same gene (here genome fragment) to appear multiply in the chromosome, then due to high fitness the whole chromosome will be flooded by same fragments. We therefore do not allow multiple copy of the same fragment. To ensure that, we used order-based crossover (OX) often used for solving TSP [30]. In OX, *offspring 1* directly copies genes from *parent 1* from the crossover point to the end of the *parent 1* chromosome. From the beginning of the *offspring 1* chromosome, the rest of the genes are copied from *parent 2* preserving the sequential relative order, thus skipping the genes already copied from *parent 1*. *Offspring 2* is constructed similarly. Here, two point crossover is also possible, but we did one point crossover in our experiments. Reciprocal exchange was used for Mutation. Simply speaking two genes are selected at random and swapped over.

### E. Chromosome Reduction Step

Through generations, chromosomes bring individual fragments with high similarity to adjacent positions by evaluation function and selection. We use this tendency to form contigs efficiently and reorganize array of genes in the chromosome in two stages, filtering stage and combining stage. We called this Chromosome Reduction Step (CRed) :

**Filtering stage :** First of all, we set a parameter $T_f$ which decides at what intervals the filtering would be executed, i.e., filtering takes place every time the generations number is a multiple of $T_f$. The genes (fragments) which are contained within contigs already formed in the best chromosome are marked. Those fragments are deleted from all the chromosomes as well as gene-table. Corresponding contigs are stored in "contig pool". We need to set up $T_f$ value properly because if we set up $T_f$ low, filtering will start even when long contigs are not yet formed inside the best chromosome. Then filtering computation will be a waste. Thus, an alternative could be initiating the filtering stage only when reasonably long new contigs are formed in the best chromosome. In our experiment, for simplicity, we fixed $T_f$. The individual chromosomes are now composed of the remaining genes, and the genetic search continues. Number of genes in the chromosome decreases gradually every time filtering is done, making the genetic search more efficient.

When decreasing until the length of the chromosome reaches ratio $r_c$ set in advance, combining stage is happened.

**Combining stage :** When a new contig is added to the contig pool, we try to combine it with the existing fragments (or contigs), if possible, to make longer contigs. Once a longer contig is formed, further genes (genome fragments) could be shed off from the chromosomes the way it is done in the filtering stage. As the contigs become longer and chromosomes shorter we can run GA more efficiently. After every combining stage, the user could check whether the available results are good enough (long enough) for her/his purpose. If not, the genetic search continues. The flow of the algorithm is shown in Fig.5, where $g$ is the number of generation.

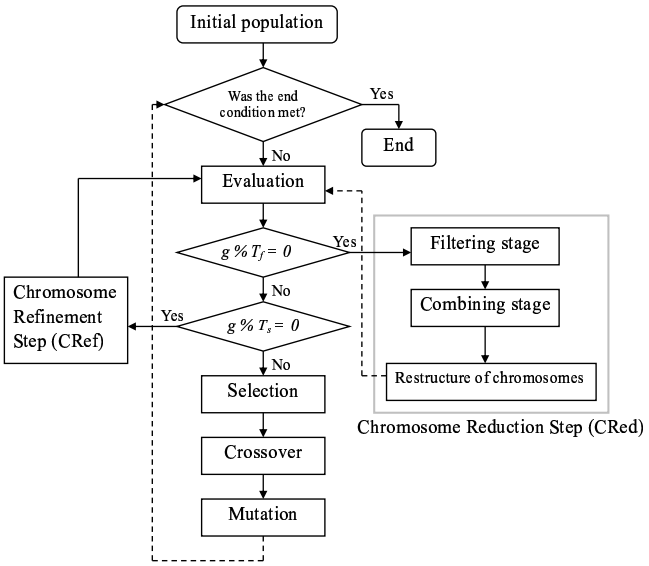


Fig. 5. Algorithm including CRed and CRef

### F. Heuristic Method — Chromosome Refinement Step

We proposed heuristic method for efficient scaffolding because scaffolds was not able to be formed with CRed alone well even though it explained by the following experiment result . There are four patterns as showing in Fig.6 and one compound pattern when two fragments have overlap.

1) overlap at the tail-part of fragment A and the beginning of fragment B
2) overlap at the tail-part of two fragments
3) overlap at the beginning of two fragment
4) overlap at the beginning of fragment A and the end of fragment B.
5) overlap at both beginning and end.

If two fragments have overlap of type 4, we swap the positions of the two fragments. With this, the positions of fragments in chromosome are arranged to correspond to there positions in the original genome. This refinement of greedy mutation takes place every $T_s$ generations, and operate only on best $n_s$ chromosomes. These two steps of CRed and CRef improve both the efficiency and quality of result of our genetic search.

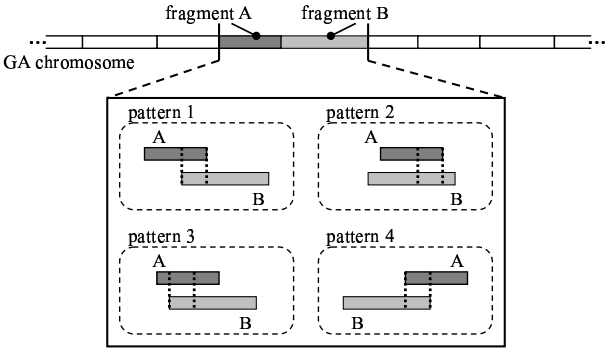


Fig. 6. Matching pattern of two fragments

## IV. Experiments and Result

### A. Experimental Set-up

The initial experiments to test the validity of our proposed algorithms were done on genomes created artificially. The test cases were scaled down, so that we can complete test runs in quick intervals. In our experiments, we used lower values (around 4X) of coverage. This made the assembling task much more difficult. But on the other hand, due to less number of fragment the execution time of the algorithms was faster.

*1) Preliminary Experiment:* This is the first set of toy experiments we did to test the possibility of solving genome sequencing problem using genetic algorithm, the way we approached. First, a genome of 1,000 bp a random sequence of A, C, G, T was constructed. We then made 10 clone copies each of which were fragmented to 10 pieces of length 100 bp + $\delta$, where $\delta$ had uniform distribution from -20 bp to +20 bp. Thus 100 such fragments were created. Due to short length of the fragments, we had groups that contain ten fragments with high similarity. The total bases of reads is 8X of original DNA, where each read was set at 40bp. These approximately 100 fragments were arranged at random to form a chromosome of GA, and the genes (fragments) inside the chromosomes were optimally ordered using Simple GA. We set the probability of crossover and mutation at 0.5 and 0.005.

*2) Results:* After 30,000 generations, fragments with high similarity were assembled together, and we succeeded to form contigs covering 98% of the original DNA on an average in each of the 20 trials. Table I shows the number, length, and other information about contigs.

On many occasions two or more contigs with large common base sequences remain in the chromosomes separated. It is possible to solve it by running many more generations of GA, it would take long time. It would be much more efficient to merge them manually. We used this experience to add the CRed step in our proposed algorithm.

TABLE I

RESULT OF PRE-EXPERIMENT

| | Average | Max | Min |
|---|---|---|---|
| Number of contigs | 22 | 25 | 18 |
| Length of contigs | 79 | 134 | 41 |
| Total size | 1872 | 1968 | 1722 |
| Percent genome covered | 98 | 100 | 89 |
| Error | 1 | 2 | 0 |

We scaled down our experiment size, and therefore our read size was also short. It is felt that with longer read lengths the efficiency would improve. In summary, if some of the domain knowledge is used to tune the chromosomes, the GA would be an efficient way for fragment sequencing.

### *B. Performance Comparison of the Proposed Algorithm with Standard GA*

From the experiments discussed in the previous section 1.1, we added steps for heuristic tuning of chromosome as stated in section Ⅲ. In this section we compare the performances and see the improvements of the proposed algorithm. In the previous experiment, though fragments were grouped together it was difficult to put them in proper order just by crossover and selection. If the fragments with high similarity are combined as contigs, and then contigs are mutually compared, the efficiency of fragment assembly would improve. In the Chromosome Reduction Step we do combine fragment to form contigs manually. Moreover we do add greedy mutation to locally sequence fragments to improve sequencing efficiency.

*1) Experimental Set-up:* As before, we scaled down the actual problem so that run time is reduced. The test genome data were of length 10,000 bp, consisting of a random array of four bases A, G, C, T. The length of fragments were set from 300 bp to 500 bp. We used 20 cloned genomes. Therefore, there were about 500 fragments. “read” was set to 40bp. The total bases of read was only 4X of the original test DNA. These values were decided referring to [26] and [21]. The crossover probability and the mutation probability were set to 0.5 and 0.01 respectively. We ran the GA up to 100,000 generations. Moreover, the parameter $T_f$, at which CRed was initiated, was set at 100.

*2) Experimental Results:* To evaluate our algorithm, we used the ratio of the contigs, lengths that had been finally obtained and the original test DNA. If the plural contig showed the same range, then only the longer contig was used to calculate. The number of contigs where a wrong base sequence had been constructed was counted as error. The above values, the obtained numbers of contigs, and their lengths are shown in Table.II. Both SGA and the proposed GA were ran for equal number of generations.

By genetic reassembling, we were able to form longer sequence contigs as shown in Fig.7. Fragments with high similarity ware gradually as the generation advanced. Especially, it shows that the proposed GA performs much better than SGA. Finally, we obtained contigs that covered about 70% of the original genome. Once a good chromosome is formed, its information is used to improve the quality of the rest of the chromosomes by detecting fragments which are no more necessary. This is achieved with filtering stage. However, our proposed GA still lacks the power and is scarcely able to form scaffolds. The maximum length of contig was 262 bp, and this corresponds to only 2.62% of original DNA. Yet, many contigs of similar sizes were obtained. Though further generation would produce longer contigs, we felt that we need to incorporate some means to facilitate scaffolding to quickly amalgamate contigs.

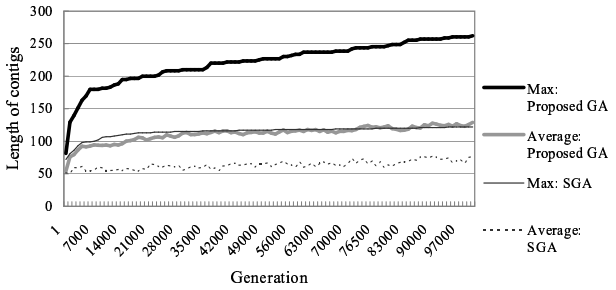


Fig. 7. SGA vs Proposed GA

### *C. Experiment using The Data of Wolbachia Genome Sequence*

We used *Wolbachia* genome sequence which is a real genome data. The actual genome sequence has more tandem repeats, but we used a smaller test genome data. We verified that it is possible to assemble fragments by the proposal technique for real genome sequence data. We compared the results obtained by our algorithm with the results using existing assemblers in *Wolbachia* genome project.

*1) Wolbachia Genome Project: Wolbachia* is a kind of bacteria and first microscopical organism monitored for horizontal gene transfer to multicellular organism. It has attracted attention of many researchers because it would be helpful in revealing the evolution of virus. Table.III is the results from existing assemblers in *Wolbachia* genome project. The of the genome size was 1.26 Mbp. The result of total size of Celera assembler using trimmed data is the nearest to the actual value and error was zero. However TIGR assembler could form the longest average contigs [31].

*2) Experimental Set-up:* We use only a small part of *Wolbachia* genome sequence to reduce computation time. It was registered in GenBank of NCBI and decoded by TIGR assembler. The part used by us is “rpoBC”, which is a gene that codes for protein and the number of bases is 8,514 bp. The details are shown in Table.IV.

We made 20 copy this genome sequence and splited fragments of length 300 bp to 500 bp. Thus, the average fragment length are about 400 and “read” is set to 40 bp. Thus the coverage is 4X. The GA is run for 1,000,000

TABLE II

RESULTS OF COMPARISON OF SGA WITH PROPOSED ALGORITHM

| | SGA | | | Proposed GA | | |
|---|---|---|---|---|---|---|
| | Average | Max | Min | Average | Max | Min |
| Number of contigs | 97 | 129 | 67 | 172 | 210 | 165 |
| Length of contigs | 76 | 122 | 51 | 130 | 262 | 54 |
| Total size | 31659 | 35282 | 28623 | 19679 | 26121 | 17682 |
| Percent genome covered | 58 | 62 | 46 | 65 | 71 | 62 |
| Error | 4 | 8 | 0 | 6 | 9 | 4 |

TABLE III

COMPARISON OF SHOTGUN SEQUENCE DATA FROM THE *Wolbachia* GENOME PROJECT [31]

| Assembler | TIGR Assembler | Celera Assembler | Celera trimmed data |
|---|---|---|---|
| Number of contigs | 76 | 220 | 101 |
| Average contig length | 16.8kbp | 6.3kbp | 12.5kbp |
| Total size | 1.28Mbp | 1.39Mbp | 1.26Mbp |
| Percent genome covered | 93.1 | 99.1 | 98.4 |
| Error | 2 | 1 | 0 |

TABLE IV

FURTHER DETAILS ABOUT *Wolbachia* GENOME DATA

| *Wolbachia* endosymbiont of Drosophila melanogaster | |
|---|---|
| whole genome | rpoBC |
| NCBI RefSeq. : NC_ 002978 | gene ID : 2738525 |
| GenBank : AE017196 | Locus tag : WD0024 |
| Length : 1,267,782 bp | Length : 8,514 bp |

generations. The crossover rate and the mutation rate, are set to 0.5 and 0.01 respectively. $T_f$ is set to 100 in filtering stage. CRef takes place at 100 generations ($T_s = 100$), just after filtering stage is executed and operates top 10 chromosomes ($n_s = 10$).

*3) Results:* The results of experiment using Wolbachia rpoBC genome data are shown in Table.V.

TABLE V

RESULTS OF EXPERIMENT USING *Wolbachia* GENOME DATA

| | Average | Max | Min |
|---|---|---|---|
| Number of contigs | 163 | 181 | 156 |
| Length of contigs | 158 | 301 | 52 |
| Total size | 18512 | 24557 | 15987 |
| Percent genome covered | 67 | 82 | 58 |
| Error | 8 | 13 | 5 |

Because we extended the number of generation, the contig length and the proportion of genome covered was improved than the previous experiments with 30,000 generation. However, the error is increased because the real-life genome has many tandem repeats. This problem could be solved by incorporating domain knowledge in the algorithm. In addition, fragments that remains in the chromosome after filtering stage, could be used for scaffolding and sequencing contigs. This is not yet incorporated in our algorithm. Yet, the results in Fig.8 shows that fitness of chromosome increases even after 1,000,000 generations. We can therefore expect to achieve better results if the algorithm is run further.

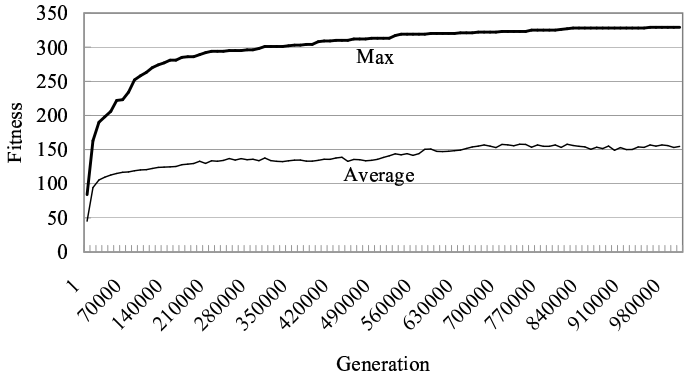


Fig. 8. Transition of fitness

Table.VI shows the comparison of the ratio of average contig length. We compared ratios of the average contig length to the length of the original genome. Our proposed technique could form long contigs on an average (though the results are based on experiments of different complexities). Even though our algorithm has obvious scope of improvements, it is able to deliver good results and is expected to be an useful tool.

As already mentioned, due to lack of adequate computational support, we scaled down the problem, and reduced the "read" length as well as coverage. This hampered the progress of search as well as the quality of the result. conditions of the actual genome analysis by the limit of our experimental environment.

TABLE VI

COMPARISON OF THE RATIO OF AVERAGE CONTIG LENGTH

| | average contig length (%) |
|---|---|
| Proposed technique | 1.86 |
| TIGR assembler | 1.33 |
| Celera assembler | 0.5 |
| Celera trimmed data | 0.99 |

## V. CONCLUSION

We proposed a genetic algorithm based approach to assemble DNA fragments to find the genome sequence. To improve the efficiency of GA for fragment assembly, we added two ideas, Chromosome Reduction step (CRed) and Chromosome Refinement step (CRef). As a result, both the speed and quality of result of assembling improved. We could obtain 80% of the Wolbachia genome sequence. The ratio of average contig-lengths compared to the whole genome length were higher.

There are scopes of improvement of our algorithm by incorporating:

- scaffolding either in GA itself, or execute it a separately at regular intervals along with combining stage.
- strategy to avoid errors due to tandem repeats.

For the above two, we can use knowledge and strategies already existing in other assembling algorithms like Celera [25].

In addition, the CRef heuristic strategy proposed here is most simple. Many modifications are possible leading to better efficiency. Another positive aspect of the genetic approach is that, one can always view the intermediate result and decide whether to continue further for better results or not.

## REFERENCES


[1] T. Gojobori, *Bioinformatics*, Springer-Verlag Tokyo, 2003, ISBN 4431709223

[2] J. C. Neil and P. A. Pavel, *An Introduction to Bioinformatics Algorithms*, A Bradford book, 2004, ISBN 0262101068

[3] P. A. Benjamin, *Genetics - A Conceptual Approach*, W.H.Freeman and Company, 2005, ISBN 0716788810

[4] P. P. Vaidyanathan, "Genomics and Proteomics : A Signal Processor's Tour," IEEE Circuits and Systems Magazine, pp.6-28, 2005

[5] M. Pop, "Shotgun Sequence Assembly," Advances in *Computers* vol.60, pp.194-248, 2004

[6] M. T. Tammi, "The Principles of Shotgun Sequencing and Automates Fragment Assembly," Center for Genomics and Bioinformatics, Karolinska Institute, Stockholm, Sweden, 2003

[7] S. Kim, "A Survey of Computational Techniques for Genome Sequencing," Project Report supported by Korea Institute of Science and Technology Information, 2002

[8] F. Sanger, A. Coulson, D. Hill and G. Petersen, "Nucleotide sequence of bacteriophage lambda DNA," J. Mol. Biol. ,162, 729-773, 1982

[9] R. D. Fleischmann, et al., "Whole-genome random sequencing and assembly of Haemophilus influenzae Rd," *Science* vol.269, Issue 5223, pp.496-512, 1995

[10] E. W. Myers, et al., "A Whole-Genome Assembly of Drosophila," *Science* vol.287, pp.2196-2204, 2000

[11] E. S. Lander, L. M. Linton, B. Birren, et al., "Initial sequencing and analysis of the human genome," *Nature* 409, pp.860-921, 2001

[12] G. G. Sutton, et al., "TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects," *Genome Science and Technology* vol.1, pp.9-19, 1995

[13] D. Gordon, C. Abajain, and P. Green, "Consed : A Graphical Tool for Sequence Finishing," *Genome Research* 8, pp.195-202, 1998

[14] X. Huang and A. Madan, "CAP3 : A DNA Sequence Assembly Program," *Genomics* vol.9, No.9, pp.868-877, 1999

[15] S. Batzoglou, D. B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J. P. Mesirov and E. S. Lander, "ARACHNE : a whole-genome shotgun assembler," *Genome Research* 12, pp.177-189, 2002

[16] S. Kim and A. M. Segre, "AMASS : A Structured Pattern Matching Approach to Shotgun Sequence Assembly," *Journal of Computational Biology* vol.6 (4), 1999

[17] P. A. Pevzner, H. Tang and M. S. Waterman, "An Eulerian path approach to DNA fragment assembly," Proceedings of the National Academy of Sciences of the USA, 98 (17), pp.9748-9753, 2001

[18] P. A. Pevzner, H. Tang and M. S. Waterman, "A New Approach to Fragment Assembly in DNA Sequencing," Proceedings of The 5th Annual International Conference on Computational Molecular Biology (RECOMB 2001), pp.256-267, Canada. ACM Press. 2001

[19] P. Green, "Phrap Documentation : Algorithms," Phred/Phrap/Consed System Home Page, *http://www.phrap.org*(current Feb. 2006)

[20] 454 Life Sciences, *http://www.454.com/*(current Feb. 2006)

[21] R. J. Parsons, S. Forrest and C. Burks, "Genetic Algorithms, Operators, and DNA Fragment Assembly," *Machine Learning* 21, pp.11-33, 1995

[22] E. Alba, G. Luque and S. Khuri, "Assembling DNA Fragments with Parallel Algorithms," *IEEE Congress on Evolutionary Computation* 2005 vol.1, pp.57-64, 2005

[23] J. Rebecca and M. E. Johnson, "DNA Fragment Assembly and Genetic Algorithms – New Results and Puzzling Insights," appeared in the Proceedings of the 3rd International Conference on Intelligent Systems in Molecular Biology, AAAI Press, pp. 277-284, 1995

[24] H. Kitano, et al., *Genetic Algorithm*, Sangyo Tosyo, 1993, ISBN 4782851367

[25] J. D. Venter, et al., "The Sequence of Human Genome," *Science* vol.291, Issue 5507, pp.1304-1351, 2001

[26] K. Mita, et al., "The Genome Sequence of Silkworm, Bombyx mori," *DNA Research* 11(1), pp.27-35, 2004

[27] M. Machida, et al., "Genome Sequencing and analysis of Aspergillus oryzae," *Nature* 438, pp.1157-1161, 2005

[28] J. Setubal and J. Meidanis, *introduction to Computational Molecular Biology*, PWS Publishing Company, 1997, ISBN 0534952623

[29] G. B. Fogel and D. W. Corne, *Evolutionary Computation in Bioinformatics*, Morgan Kaufmann, 2002, ISBN 1558607978

[30] Z. Michalewicz, *Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1999, ISBN 3540606769

[31] M. Pop, S. L. Salzberg and M. Shumway, "Genome Sequence Assembly, Algorithms and Issues," *Computers* 35(7), pp.47-58, 2002