Abstract—Shotgun sequencing is the state-of-the-art to decode genome sequence. However, this technique needs a lot of fragments. Combining those fragments correctly requires enormous computational cost. In our previous work we have shown how Genetic Algorithm (GA) could solve this problem efficiently. In this work, we added two heuristic ideas with GA to make it more efficient. One is Chromosome Reduction (CRed) step which shortens the length of the chromosomes, participating in genetic search, to improve the efficiency. The other is Chromosome Refinement (CRef) step which is a greedy heuristics, rearranging the bits using domain knowledge, to locally improve the fitness of chromosomes. With this hybridization and simple scaffold list, we could obtain longer contigs and scaffolds using GA. We experimented using three actual genome data to test our algorithm. We succeed in restructuring contigs covering about 90% of target genome sequences, and assembling about 500 ∼ 1,000 fragments into 3 ∼ 11 scaffolds. All the experiments were done using common desktop machines.

I. INTRODUCTION

Though the post-genome research attracts a great deal of attention by the achievement of decoding the human genome [1], still there are many important genome sequences of living things which yet to be decoded. These would bring benefits to our lives, like the recent decoding of populus trichocarpa genome which is first tree genome sequenced in 2006 [2]. Recently, over 300 genome sequences, which were decoded completely, were registered with NCBI (The National Center for Biotechnology Information) database and another 750 are in line [3]. However, whereas the range of genome sequences are large, from small viruses (a few thousand nucleotides) to large mammals (∼ 3 giga nucleotides), current technology could not read all base-pairs at the same time due to physical limits. Therefore we must cut the target genome into small fragments, read base sequence of those fragments and reassemble the original genome using information of the overlapping portions. This procedure is called shotgun sequencing and is the commonly used method [4] [9] [10]. To reconstruct the target genome, lots of fragments are needed and assembling those accurately is a NP-hard problem. Several deterministic algorithms based on graph-theory, and greedy heuristic algorithms are proposed. But they are extremely computationally involved methods, and need large scale distributed processing network which is very costly. Worldwide only a few such installations are available. The main motivation of this work is to find an efficient fragmentation assembly algorithm that could run on cheap computers, yet be able to find nearly correct draft sequences.

The rest of the paper is organized in the following sections. In section 2 we gave a brief survey of the existing fragment assembly methods. In section 3, shotgun sequencing and problems of the existing techniques are briefly explained. Section 4 is devoted to explaining the proposed algorithm. In section 5, we state the results of the experiments using three actual genome sequences and discussed about them. Conclusion is in section 6.

II. EXISTING FRAGMENT ASSEMBLY METHODS

To read a genome sequence, Sanger sequencing is the well known [11] and most commonly used. It uses Gel electrophoresis that facilitates reading the bases due to reaction of the fluorescent dye staining different bases differently. Sanger sequencing can read on an average of a mere 500 to 800 base pairs. During 80's shotgun sequencing (fragmenting the genome and assembling) based on Sanger method could successfully sequence up to 10 Kbps (kilo base pairs), and by 1990 it could sequence segments up to 40 Kbps. In 1995, Fleischmann et al. [12] could assemble the ∼1,800 Kbps long H. Influenzae bacterium, and in 2000 Myers et al. [13] was able to assemble ∼130 Mbps long Drosophila genome. By 2001, Lander et al. [14] presented an initial sequencing of human genome of ∼3.5 Gbps length. Many existing fragment assembly systems read the fragment base-sequence by Sanger technique and reconstruct the original genome sequence with their proprietary assembling algorithms. Many assembling algorithms were proposed, the important ones being TIGR assembler [15], Consed [16], RAME [32], Cerela Assembler [13], CAP3 [17], ARACHNE [18], AMASS [19], EULER [20], Phrap [21]. A good survey of many of these algorithms is available in [10].

During last ten years a few works were reported to use genetic algorithm (GA) to solve fragment assembling problem. Many techniques use GA, based on Paesons’s works [25] [26]. Recent works [27] [28] used distributed GA. [29] also based on standard GA, and so is our previous [30]. 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 chromosomes and thereby the complexity of the search-space, and (2) Chromosome Refinement Step (CRef) to locally improve the fitness of chromosomes by a kind of greedy mutation using domain knowledge. And we aim to achieve a series of process leading up to scaffolding by only running GA.

III. SHOTGUN SEQUENCING METHOD AND ITS PRESENT STATUS

A. Shotgun Sequencing

In this section, we explain assembling based on Sanger sequencing. To decode a long DNA sequence we need to split it up, read the individual fragments and then assemble them in correct sequence to reconstruct the target DNA. 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. It is called Clone-by-clone shotgun sequencing, and is often used in the international genome project to obtain complete genome sequence.

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 too complex due to repeat stretches in the genome. Yet, in 2000 Myers et al. successfully sequenced the fruit fly genome. Yet, in 2000 Myers et al. successfully sequenced too heavy, and too complex due to repeat stretches in the genome. It was thought to be computationally too heavy, and too complex due to repeat stretches in the genome. And we aim to achieve a series of process leading up to scaffolding by only running GA.

III. SHOTGUN SEQUENCING METHOD AND ITS PRESENT STATUS

A. Shotgun Sequencing

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 too complex due to repeat stretches in the genome. Yet, in 2000 Myers et al. successfully sequenced the fruit fly genome. Yet, in 2000 Myers et al. successfully sequenced the fruit fly drosophila genome of length ≈ 125 Mbps using WGSS [13], which ushered in and established WGSS 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 limitations in mind, WGSS was used in the determination of draft human genome in 2001 by Celera Genomics [1]. In recent years, WGSS also decoded several genomes of living things, e.g., the Silkworm by Mita in 2004 [32], the Aspergillus oryzae by Machida in 2005 [33], the chimpanzee by Mikkelsen in 2005 [34], the honeybee Apis mellifera by Weinstock in 2006 [35], and the Sea Urchin Strongylocentrotus purpuratus by Sodergren in 2006 [36].

Our target is similar, to create the sequence based on WGSS, doing the assembling part using GA. The algorithm will bring out longer and longer contigs with shorter and shorter gaps, as it continues running. Once the target result is reached, we would stop there instead of further searching. 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 steps - one is the biological part of cloning, fragmenting, and reading, and the next step is the computational one 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. Thus the fragments are of different lengths. Fragments that are too large or too small are then discarded. Of the remaining fragments, i.e., those used for assembling, the length of short ones are about 2kbp, and the long ones are about 10kbp [4]. 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 (because they are inserted in the phage), and the set of inserts with similar size, 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 a 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](image)

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 the redundancy (the copiousness) of the fragment read data. It is defined as the ratio of the number of bases read from fragments to the length of the source DNA [37].

\[
Coverage = \frac{\sum_{i=1}^{n} \text{matepairs of fragment}_i}{\text{target_genome_length}}
\]

(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~ 10X). If coverage is high, the probability of covering original genome is higher and the accuracy is improved. However, the number of fragments involved in the assembling and therefore the computational complexity also increases. In practice, to sequence large genomes, hundreds of thousands to tens of millions of fragments are used for assembling. 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. At the bottom of Fig.2, we have shown 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, especially for short read lengths, as there could be repetitive short sequences in the original genome.

![Formation of contigs](image1)

**Fig. 2. Formation of contigs**

The existing techniques compare all fragments for overlap detection using parallel and 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” [1]. Obviously, such computational support is far 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.

![Scaffolding](image2)

**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 the important assemblers are using this paradigm. For example, Celera assembler [1] employs scaffolding algorithm based on graph theory using mate pairs. TIGR assembler [15] 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. 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 in progress of their researches 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, or whether a particular sequence is present in the genome or not, and not the sequence of the whole genome.

Genetic Algorithm (GA) is already a proven robust algorithm for graph searching and many other combinatorial NP-hard problems. There are a whole bunch of works on multiple alignment technique, that use evolutionary or generic algorithm [38] [39]. But very few works used GA to solve the whole genome assembly problem [28] [26] [27] [29].

Our proposed GA approach is standard genetic algorithm like the one proposed by Parsons et al. [25]. 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 complexity of the search-space, and (2) Chromosome Refinement Step (CRef) to locally improve the fitness of chromosomes (a type of greedy mutation 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.

IV. PROPOSED FRAGMENT ASSEMBLING TECHNIQUE USING GENETIC ALGORITHM

Our proposed technique is based on standard GA coupled with two heuristic methods. One is called Chromosome Reduction Step (CRed), which reduce the search space for efficient GA optimization. Chromosome Refinement Step (CRef) is the greedy mutation to improve the accuracy of
the solution using the domain knowledge. We were able to combine the phase of overlap and scaffolding in the GA search by the way we defined the structure of the chromosome and CRed.

A. The Construction of the Chromosome

The gene of our GA chromosome are the fragments. There is the information of the two read sequences which are mate-pair and the gap of unknown length in between. The search is optimized including the information of the mate-pair. Therefore the scaffolds are generated when contigs are found.

The fragments generated by Shotgun sequencing method are labeled in serial numbers, 1 to \( N \), where, \( N \) is the total number of fragments. The read information corresponding to different fragments is stored in Fragment Data Table (FDT). The chromosome is composed of all these \( N \) fragments sequenced in random. Thus a chromosome is actually a permutation of numbers 1 to \( N \), which are labels of different fragments. A number of such chromosomes, equal to the population size is created. 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 (genes of the chromosome). In the same way, the information of mate-pair is also retained without break.

Here we set a threshold value \( mp \) of similarity to judge the extent of similarity between two fragments. If two fragments \( i \) and \( i + 1 \) have the same sequence accounting for more than or equal to \( mp \% \) of the read, similarity \((i, i+1)\) score is 1, and otherwise it is 0. similarity \((i, i+1)\) could have discrete values from 0 through 2. Because each fragment has two reads, similarity is 2 if two ‘read’s on both sides of the fragments matched. If the ‘read’ on only one side matches, the score is 1. If we set \( mp \) low, the mis-assembling probability increases. On the other hand, if we set \( mp \) high, the progress of GA optimization will be very slow because it would be almost impossible to find similar fragments placed next to each other.

C. Selection, Crossover and Mutation

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 selection, which also helps in avoiding local minimum. We also use elitist preservation.

If we allow same gene (here genome fragment) to appear multiply in the chromosome (which is possible as the result of a crossover), then due to high degree of match and consequently high fitness, the whole chromosome will be flooded with the same fragment. We therefore do not allow multiple copies of the same fragment. To ensure that, we used order-based crossover OX and reciprocal exchange, often used in solving TSP [40]. In reciprocal exchange, it is possible to swap the subset if the selected gene is included in the subset. By doing so the fitness will improve because we can avoid the collapse of the subset.

D. 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 :**

Filtering stage improves the efficiency of GA optimization by finding contigs, extracting them from the chromosome and thereby shortening the length of chromosome. The procedures is as follows.

**Step 0.** \( s \leftarrow \text{coverage} + 1; \)

**Step 1.** After evaluation, search for the presence of a substring in the best chromosome that accommodates at least \( s \) fragments.

**Step 2.** If such subset/s is/are found,

Mark the fragments which are contained within the subset/s

If this is the first time filtering is successful

**Step 3.**

...
Step 3. Delete those fragments from all the chromosome as well as FDT.
Step 4. Store the extracted contig/s in “contig pool”;
Step 5. At every $t_{cd}$ generations elapses, since the last generation when contig consisting of $s$ fragments was found, then $s \leftarrow s - 1$;

Finding contigs is a computationally heavy task. Determining good intervals to check whether new contigs are formed, is tricky. It should not be too short so that the trial to find contig of proper length will mostly fail, nor should it be too long so that contigs are formed but remain in the chromosome without being extracted out for long causing inefficient genetic search with longer chromosomes. The first time, when $s$ is longest, the time taken for contig formation is longest. We set $t_{cd}$ as the interval to check if new contigs are formed. As long as we find contigs of length $s$ every $t_{cd}$ generations, $s$ is not changed. If after $t_{cd}$ generations, we can not find contig of length $s$, we limit our expectations to contigs of smaller lengths, decrementing the parameter $s \leftarrow (s - 1)$, which is done in Step 5 above.

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.

Moreover we utilized “contig pool” to structure the scaffolds. However we did not deploy any special strategy, just used the data in the contig pool. Thus, scaffolds too are formed by running GA only.

In filtering stage of CRed, the fragments in the substring extracted from the chromosome may join to one end of an existing contig, or it may join two contigs on two sides to form a very long contig. Information of all contigs in contig pool is added to Contig Data Table (CDT). And information about their relationship, if any, are added to the Scaffold Data Table (SDT). Thus, SDT holds the information about the label of related contigs and their relative positions. Every time combining stage starts, new contigs are compared with existing contigs and combined if the similarity is high. CDT and SDT are renewed after that. Using simple user interface, the formation of contigs and scaffolds can be visualized and it is possible to be manipulated by an expert if necessary.

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.

E. Heuristic Method — Chromosome Refinement Step

Instead of depending on genetic search alone, we add a heuristic step to facilitate scaffolding more efficiently. This is a simple and fast heuristic named CRef as explained below.

When two fragments A and B are sequentially positioned in a chromosome due to overlap, the following overlap patterns, as shown in Fig.6, are possible.

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 (not shown in Fig.6).

If two fragments have overlap of type 4, we swap the positions of the two fragments. With this, the positions of fragments in GA chromosome are arranged to correspond to their positions in the original genome. And though we could compare adjoining two fragments only in the evaluation, it is possible to find the fragments with high similarity which are next adjoining fragment by CRef. The specific procedures are as follows. Here, $g$ is the number of generations and $N$ is the total number of gene in the chromosome.

Step 0. set $f_{cr}$ and $rc$ ;
Step 1. If $g \% t_{cd} = 0$,
For $i = 1, \ldots, N$,
...
Search fragment-matching as type 4 according to \( f_{sf} \) fragments in the top \( rc \) chromosome;

Step 2. If type 4 matching is found,

Compare matching positions of those fragments and rearrange them;

If \( f_{sf} \) and \( rc \) are set high, the computational cost increases, though the fitness is improved more. Therefore we need some trade-off. With these two steps of CRed and CRef, both the efficiency and quality of result of our genetic search greatly improved.

V. EXPERIMENTS AND RESULTS

A. Experimental Genome Data

We used three real genome sequence data to test the effectiveness of our algorithm. They are included in the NCBI database. rpoBC is the base sequence of Wolbachia’s gene and its length is 8514 bp. POBF and AMCG are used in the experiments of Parsons et. al. [25][26]. POBF is the human apolipoprotein, which is 10089 bp long. AMCG is the initial 40% of the bases from LAMCG which is the complete genome of bacteriophage lambda and its length is 20100 bp. We scaled down the experimental parameters to reduce the computation time. We cloned each sequence and fragmented them in 200 bp to 500 bp imitating the shotgun method. Each \( read \) is set to 40 bp. We set the \( coverage \) to a much lower value of 4X rather than 8X \( \sim \) 10X used in actual genome sequencing. Due to lack of redundancy of information, it is difficult to reconstruct the complete target genome sequence. The details of the experimental genome data are shown in Table I.

B. Set-up of GA Operators

Population size is set at 100 chromosomes which are generated by technique explained in the previous section. The crossover rate and the mutation rate are set at 0.8 and 0.05 respectively. In the experiment using rpoBC sequence we run genetic updating for 40 hours, similarly 50 hours for POBF experiment and 100 hours for AMCG experiment. The threshold parameter \( m_{sp} \), judging the similarity level of adjacent fragments, is set to 50%. This setting made it difficult to arrange fragments with similarities. In fact, a much lower value of 5% is generally used [1]. We couldn’t use such lower value because of scaling down of the \( read \) to only 40 bp. (even50% of 40 bp is less than 5% of actual \( reads \) of 500 to 800 bp). In spite of that, if we could get good results under our experimental settings, it would prove that our algorithm would be able to get much better results with actual fragment assembly data.

We defined \( s \) in section III.E. This parameter, which triggers the starting of CRed operation, is set at \( coverage + 1 \). When we used simple GA (without CRed or CRef) for fragment assembly with rpoBC, the improvement of the solution often stopped around 55,000 generations, the average of 20 trials is shown in Fig. 7. At that point the subset contained \( coverage + 1 \) or \( coverage + 2 \). Apparently \( coverage \) means that there will be four same base sequences if \( coverage \) is set at 4X. However there are some bias depending on the position of the part because fragments are cut randomly. Therefore we thought that we should set \( s = coverage + 1 \).

In addition, we examined the proper value of the parameter \( f_{sf} \), which is used in CRef. We set \( f_{sf} \) to 2 to 5 and the top 10 chromosomes are treated for CRef (i.e., \( rc = 10 \)) at intervals of every 100 generations (i.e., \( t_{sf} = 100 \)). rpoBC sequence is used in these experiments, and the results are shown in Table II.

In Table II, the number of improvement in the second column is incremented if the fitness improves or even remains equal by CRef. This is because the arrangement in the chromosome is better than the original chromosome even if the fitness is equal after application of CRed. And CRef sometimes breaks the subset causing decrease in the chromosome fitness. Thus fitness remains equal is actually improvement in the arrangement. The best result is obtained when \( f_{sf} = 3 \). If \( f_{sf} \) is set lower, there is a probability of breaking of large subsets of sequence by the disruptive greedy swap of CRef. However CRef also disrupts subsets because the area targeted by CRef is wide and cut across plural subsets if \( f_{sf} \) is set high. Especially at the later stage most of the subsets in the chromosome are small, because large subsets are picked up by CRed. Therefore CRed doesn’t work well because the area targeted by CRef is much wider compared to the size of subset. The size of subset depends on \( coverage \) and it doesn’t break subset even at the later stage of optimization, so much so that CRef worked well when \( f_{sf} = 3 \) which is a value near to the \( coverage \). We finally set \( s \) at 5, and \( f_{sf} \) at 3, based on the results of the experiments mentioned above.

C. Results

The contigs generated using our algorithm, for the three genome data, are shown in Table III. The improvement of contig length (for the best trial) using rpoBC data is shown in Fig. 8.

Though the complexity of assembling the target genome increases exponentially with the genome size, and our experiments were with only small length genomes,
TABLE I
EXPERIMENTAL GENOME DATA

<table>
<thead>
<tr>
<th></th>
<th>rpoBC</th>
<th>POBF</th>
<th>AMCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>8514 bp</td>
<td>10089 bp</td>
<td>20100 bp</td>
</tr>
<tr>
<td>Locus tag</td>
<td>WD0024</td>
<td>Accession No. : M15421</td>
<td>Accession No. : J02459</td>
</tr>
<tr>
<td>DNA-directed RNA polymerase, beta/beta' subunits</td>
<td>fragment : about 450</td>
<td>Human apolipoprotein B-100 mRNA, complete cds.</td>
<td>Bacteriophage lambda, complete genome (initial 40%)</td>
</tr>
<tr>
<td>(Wolbachia endosymbiont of Drosophila melanogaster)</td>
<td>fragment : about 500</td>
<td>fragment : about 1000</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II
COMPARISON OF THE PARAMETER $f_{cf}$

<table>
<thead>
<tr>
<th>$f_{cf}$</th>
<th>Number of improvement</th>
<th>Length of contig</th>
<th>Total size</th>
<th>Percent genome covered</th>
<th>Computational time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{cf} = 2$</td>
<td>Best</td>
<td>9996</td>
<td>472</td>
<td>9710</td>
<td>88.3</td>
</tr>
<tr>
<td></td>
<td>Ave.</td>
<td>9654.9</td>
<td>214.2</td>
<td>10577.9</td>
<td>80.1</td>
</tr>
<tr>
<td>$f_{cf} = 3$</td>
<td>Best</td>
<td>12945</td>
<td>531</td>
<td>9106</td>
<td>96.3</td>
</tr>
<tr>
<td></td>
<td>Ave.</td>
<td>10299.0</td>
<td>329.1</td>
<td>9854.6</td>
<td>87.1</td>
</tr>
<tr>
<td>$f_{cf} = 4$</td>
<td>Best</td>
<td>9894</td>
<td>508</td>
<td>9460</td>
<td>92.3</td>
</tr>
<tr>
<td></td>
<td>Ave.</td>
<td>9610.8</td>
<td>303.4</td>
<td>10286.1</td>
<td>86.5</td>
</tr>
<tr>
<td>$f_{cf} = 5$</td>
<td>Best</td>
<td>9600</td>
<td>497</td>
<td>10044</td>
<td>87.3</td>
</tr>
<tr>
<td></td>
<td>Ave.</td>
<td>9398.6</td>
<td>291.8</td>
<td>11429.2</td>
<td>85.1</td>
</tr>
</tbody>
</table>

TABLE III
RESULTS ABOUT CONTIG

<table>
<thead>
<tr>
<th></th>
<th>Number of contigs</th>
<th>Length of contigs</th>
<th>Total size</th>
<th>Percent genome covered</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpoBC : Best</td>
<td>20</td>
<td>531</td>
<td>9106</td>
<td>96.3</td>
<td>0</td>
</tr>
<tr>
<td>Ave.</td>
<td>27.7</td>
<td>329.1</td>
<td>9854.6</td>
<td>87.1</td>
<td>0</td>
</tr>
<tr>
<td>POBF : Best</td>
<td>27</td>
<td>702</td>
<td>12436</td>
<td>94.7</td>
<td>0</td>
</tr>
<tr>
<td>Ave.</td>
<td>30.1</td>
<td>376.2</td>
<td>14105.8</td>
<td>86.5</td>
<td>0</td>
</tr>
<tr>
<td>AMCG : Best</td>
<td>31</td>
<td>1590</td>
<td>27854</td>
<td>89.4</td>
<td>0</td>
</tr>
<tr>
<td>Ave.</td>
<td>45.1</td>
<td>681.8</td>
<td>32081.2</td>
<td>81.1</td>
<td>0</td>
</tr>
</tbody>
</table>

If looked carefully, one can see small step increments in the transition curve of Fig.8. This means that CRed could update the parameter $s$ properly to avoid stagnating the growth of the subsets. The length of contigs was longer again because updating of $s$ lead to finding smaller subsets. For this reason, CRed is a useful technique which is able to adjust to the status of the chromosome and work efficiently.

The scaffolds results we got are shown in Table. IV. These results are impressive too. We got long scaffolds covering almost half the length of the target genome sequence in the best results. The scaffolds included several contigs too. This results were achieved, without any extra computation, by design of of our GA chromosome, evaluation function and SDT, the simple database. In genome sequencing, not only generation of contigs but also structuring of scaffolds is important. Therefore our proposed algorithm, which takes care of both, is realistic fragment assembling method.

VI. CONCLUSION

We proposed a genetic algorithm based approach to assemble DNA fragments to construct the genome sequence. We added two ideas, Chromosome Reduction step (CRed) and Chromosome Refinement step (CRef), to improve the
efficiency of GA optimization for fragment assembly. And with the help of simple database we could achieve scaffold formation. As a result, we got longer contigs and scaffolds just by running our GA. We could obtain about 90% of the three target genome sequences we experimented and assemble many fragments to scaffolds.

Our results suffered due to low coverage in the experiments we did (only 4X instead of more practical approx10X). More fragments covering the same part of the sequence is preferred for better accuracy. Also, introducing biological knowledge is helpful for effective fragment assembling and improving its accuracy, which we plan to link with our genetic search, in CRed and CRef. In amino acid and protein formation, there are some distinct rules in the alignment of bases. Using that pattern knowledge we could decrease the computational cost during the similarity measurement using dynamic programming. We plan to include domain knowledge to design more efficient dynamic programming specific for our purpose.

Though CRed step is proposed for fragment assembly problem, it is applicable to similar problems like clustering, path search and other combinatorial optimization.

REFERENCES

[34] T. S. Mikkelsen, et al., “Initial sequence of the chimpanzee genome and comparison with the human genome,” Nature 437, pp.69-87, 2005