Genome assembly composition of the String "ACGT" array: A review of data Structure accuracy and performance challenges

Sherif Magdy Mohamed Abdelaziz Barakat [Corresp., 1] , **Roselina Sallehuddin** [1] , **Siti Sophiayati Yuhaniz** [2] , **Raja Farhana R. Khairuddin** [3] , **Yasir Mahmood** [4]

[1] Computer Science, School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia

[2] Advanced Informatics Department, Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur, Kuala Lumpur, Malaysia

[3] Department of Biology, Universiti Pendidikan Sultan Idris Tanjung, Malim, Malim, Malaysia

[4] Faculty of Information Technology, The University of Lahore, Lahore, Lahore, Pakistan

Corresponding Author: Sherif Magdy Mohamed Abdelaziz Barakat
Email address: sherifmagdy.barakat@gmail.com

Background. In this era, biologists have been increasingly interested in interpreting the genome sequence. Unfortunately, the current sequencing technology is limited to providing the genome as a massive number of short strings (reads). Computer algorithms for genome assembly construct the entire genome from reads in two approaches. De novo approach concatenates the reads based on the exact match between their suffix-prefix (overlapping). Reference-guided approach orders the reads based on their offsets in a well-known reference genome (reads alignment). Repetitive sequences in short reads(repeats) present technical ambiguity, making the algorithm unable to distinguish the reads and assemble them incorrectly (misassembly). On the other hand, the massive number of reads causes a big assembly performance challenge. **Method.** The repeat identification method was introduced for misassembly by prior identification of repetitive sequences, creating a repeat knowledge base to reduce ambiguity during the assembly process, thus enhancing the accuracy of the assembled genome. Also, hybridization between assembly approaches resulted in a lower misassembly degree with the aid of the reference genome if it exists. On the other hand, assembly performance is optimized through data structure indexing or parallelization. This paper's primary aim and contribution are to support the researchers through an extensive review to ease other researchers' search for genome assembly studies. We highlighted the most recent developments and limitations in genome assembly accuracy and performance optimization. **Results.** As a result of this review, we found that most repeat identification methods can detect only a specific length of the repeat, while repeat in biology varies. On the other hand, hybridization between de novo and reference-guided approach lacks any repeat resolution. In addition, it is too time-intensive. Regardless of the tremendous work on genome assembly performance optimization, no current method fits the optimization of the hybrid assembly approach. **Conclusion.** We suggest combining multiple repeat identification methods to enhance the accuracy of identifying the repeats as an initial step to the assembly hybrid approach and combining genome indexing with parallelization for better optimization of its performance.

1 # Genome Assembly Composition of the String "ACGT"
2 # Array: A Review of Data Structure Accuracy and
3 # Performance Challenges

4 Sherif Magdy Mohamed Abdelaziz Barakat[1], Roselina Sallehuddin[1], Siti Sophiayati Yuhaniz[2],
5 Raja Farhana R. Khairuddin[3], Yasir Mahmood[2]
6
7
8 [1] Computing Department, School of Computing- Faculty of Engineering, Universiti Teknologi
9 Malaysia Skudai, Joho, Malaysia
10 [2] Advanced Informatics Department, Razak Faculty of Technology and Informatics- Universiti
11 Teknologi Malaysia, Kuala Lumpur, Malaysia
12 [3] Department of Biology, Universiti Pendidikan Sultan Idris Tanjung, Malim, Malaysia
13
14 Corresponding Author:
15 Sherif Barakat[1]
16 UTM, Jalan Iman, 81310 Skudai, Johor
17 Email address: sherifmagdy.barakat@gmail.com

19 ## Abstract

20 **Background.** The development of sequencing technology increases the number of genomes being
21 sequenced. However, obtaining a quality genome sequence remains a challenge in genome
22 assembly by assembling a massive number of short strings (reads) with the presence of repetitive
23 sequences (repeats). Computer algorithms for genome assembly construct the entire genome
24 from reads in two approaches. The *de novo* approach concatenates the reads based on the exact
25 match between their suffix-prefix (overlapping). Reference-guided approach orders the reads
26 based on their offsets in a well-known reference genome (reads alignment). The presence of
27 repeats extends the technical ambiguity, making the algorithm unable to distinguish the reads
28 resulting in misassembly and affecting the assembly approach accuracy. On the other hand, the
29 massive number of reads causes a big assembly performance challenge.
30 **Method.** The repeat identification method was introduced for misassembly by prior identification
31 of repetitive sequences, creating a repeat knowledge base to reduce ambiguity during the
32 assembly process, thus enhancing the accuracy of the assembled genome. Also, hybridization
33 between assembly approaches resulted in a lower misassembly degree with the aid of the
34 reference genome. The assembly performance is optimized through data structure indexing and
35 parallelization. This paper's primary aim and contribution are to support the researchers through
36 an extensive review to ease other researchers' search for genome assembly studies. The study
37 also, highlighted the most recent developments and limitations in genome assembly accuracy
38 and performance optimization.
39 **Results.** Our findings show the limitations of the repeat identification methods available, which
40 only allow to detect of specific lengths of the repeat, and may not perform well when various

41 types of repeats are present in a genome. We also found that most of the hybrid assembly
42 approaches, either starting with *de novo* or reference-guided, have some limitations in handling
43 repetitive sequences as it is more computationally costly and time intensive. Although the hybrid
44 approach was found to outperform individual assembly approaches, optimizing its performance
45 remains a challenge. Also, the usage of parallelization in overlapping and reads alignment for
46 genome assembly is yet to be fully implemented in the hybrid assembly approach.
47 **Conclusion.** We suggest combining multiple repeat identification methods to enhance the
48 accuracy of identifying the repeats as an initial step to the hybrid assembly approach and
49 combining genome indexing with parallelization for better optimization of its performance.

Introduction

Repeats in the genome

56 Deoxyribonucleic acid (DNA) is the genetic material of most organisms, which is made up of a
57 chain of four chemical bases indicated by letters A, C, G, and T, and a complete set of DNA
58 sequences an organism called genome [45]. Repetitive sequences were found across all
59 kingdoms of life. More than 50% of the human genome is occupied by DNA repetitive
60 sequences. Repeat is a segment of sequence that appears multiple times in the genome in the
61 identical or near-identical form [14],[47]. The essential repeat categories in biology are
62 transposable elements (TE), and tandem repeats (TR) [1]. TEs are DNA sequences that are
63 able to copy themselves from one genome region, overwriting another region , which are
64 considered to play a major role in the genome evolution, thus, can change the structure and size
65 of genomes, as shown in Figure.1. The length of TE can be varied, such as long-terminal repeat
66 retrotransposons, the length of which generally ranges from 100 bp to 25 kb [64]. On the other
67 hand, TR is a repetitive DNA sequence, which is either known as a microsatellite for a short
68 repeat segment (1- 12 bp) or a minisatellite for a longer repeat segment (12 - 500 bp) located
69 adjacent to each other. Both types of repeats have the ability to expand their copy number and
70 change the structure and size of the genome [2],[3]. The expansion of repeat sequences can cause
71 many diseases in humans, such as Huntington disease and Kennedy disease [4]. The presence of
72 the repetitive offers crucial biological information that stores historical genome changes within
73 and between species, which should be properly handled during assembly , and cannot be treated
74 as a duplication and then truncated by data cleaning.

75

Genome Assembly in Genome Analysis Project

77 Genome analysis can be divided into three main phases; genome sequencing, genome assembly,
78 and genome annotation. In the first phase, genomic sequences from an organism will be
79 extracted and further processed through a sequencing machine, in which the sequences will be
80 fragmented into smaller fragments as an output (reads). In the genome assembly phase, all the
81 reads will be overlapped or mapped to the existing reference genome to construct the genome of

82 the organism. Finally, the assembled genome will be annotated according to their genome
83 position for coding (e.g. gene) and non-coding (e.g. intron) regions. The genome assembly phase
84 is the most computationally intensive part of genome analysis. Given such a collection of reads,
85 a genome containing all genetic information about an organism will be constructed. In the
86 genome assembly, reads(substrings) are concatenated together to construct the genome. The next
87 generation sequencing (NGS), which is one of the sequencing technology, output (reads) genome
88 reads are available in public repositories such as National Centre for Biotechnology Information
89 (NCBI) [54], and DNA Databank of Japan (DDBJ), UC Santa Cruz Genomics Institute (UCSC)
90 [55]. Genome data reads can be stored in FASTA and FASTQ files [54].
91
92 **Data Structure Approaches for Genome Assembly**
93 An exact match is considered when two matched strings are identical, as shown in Figure.2-A.
94 On the other hand, the approximate match represents the high similarity between two strings and
95 is not necessarily identical, and some differences might be there. The number of different letters
96 between similar approximate matched strings is called distance [5]. The distance can be a
97 substitution representing a letter in the target string differently from its corresponding one in the
98 reference string [43]. The distance can also be an insertion of a letter in the target string that does
99 not exist in its corresponding offset in the reference string. Another type of distance is deletion,
100 which one letter is absent in the target string compared to its corresponding offset in the
101 reference string, as shown in Figure.2-B. *De novo* is Latin word that means from scratch or new.
102 One of the genome assembly approaches is de novo approach, which concatenates overlapping
103 reads with the exact match between the suffix of a read with a prefix of another read [6].
104 Overlapping reads will be concatenated into a longer string called a contig. Similarly,
105 overlapping contigs will be concatenated into a scaffold, as shown in Figure 3.[7]. Overlapping
106 distance (length) is a significant parameter for any overlapping algorithm. Overlapping errors
107 have a small impact when sufficient distance is used, as shown in Figure.4, which suggests that
108 empirical overlapping distance should be greater than 40% from the read length [8].
109 On the other hand, the assembly is guided by known reference genome instances in the
110 reference-guided approach. Reads are mapped against the reference genome to determine their
111 orders, called the reads alignment process [46], as shown in Figure.5. The reads alignment
112 against the reference genome is based on an approximate match between the read and its
113 corresponding k-mer of the reference genome. Mer is a Latin word that means part, while k is the
114 length of this part [9]. It is not necessarily the reads exactly match the reference k-mer, because
115 reads of the target genome are slightly different from the used reference genome due to
116 individual biological variation [1].
117
118 **Limitations Of Current Sequencing Technology as the Root Cause of Genome Assembly**
119 **Challenges**
120 In 1965 Fred Sanger and colleagues sequenced the first DNA reads to produce reads slightly less
121 than one kilo-base (kb) in length. However, it was first-generation DNA sequencing. Second-
122 generation DNA sequencing or next-generation sequencing (NGS) is a massive number of tiny
123 Sanger sequencing running in parallel. In NGS, large quantities of DNA can be sequenced
124 quickly. Short-length reads are considered the first limitation in this technology, generating a big
125 challenge for assembling repetitive sequences from short reads. Recently, third-generation DNA

126 sequencing (TGS) was introduced by Stephen Quake. The main characteristic of this method is
127 generating longer-length reads. TGS is currently available, generating longer-length reads able to
128 bridge long repetitive sequences, and used to develop assemblies such as Canu and Flye [65].
129 However, the natural high error rate of up to 15% of long reads impacts the accuracy of the
130 assembly requiring higher computational complexity correction processes when processing large
131 datasets [57],[64]. The most common question is why DNA molecules cannot be sequenced as a
132 single string. Firstly, the enzyme used in the sequencing experiment naturally generates sequence
133 errors after sequencing three kilo-base [11]. Secondly, the DNA sequencing experiment is too
134 much time-consuming. Thus, sequencing the entire genome using a single sequencing machine
135 would take years to be done [12]. There is no sequencing platform that provides complete
136 coverage of the whole genome as a single string [13].In NGS, each region is sequenced many
137 times in order to ensure that genome is accurately sequenced, called coverage. Data coverage is
138 the average number of times a base of a genome is sequenced to ensure that the genome is
139 sequenced accurately. It is often expressed as (1x, 2x, 3x,…, nx) as shown in Figure.6. Target
140 genome reads from NGS are extremely massive data. For example, the number of 60X sequence
141 coverage is often mentioned in that the number of letters in the dataset is 60 times bigger than the
142 original genome size [14]. This massive data amount is considered the second limitation of NGS,
143 resulting in big performance challenges during genome assembly.

144
145

146 ## Genome Assembly Challenges

147

148 **Assembling Repeats from Short Reads Misassembly**
149 The presence of repeats in a genome has increased the complexity of accurately assembl the
150 genome from short reads. [15],[35]. In the *de novo* approach based on overlapping, algorithms
151 might concatenate reads wrongly, leading to a misassembly simply because they have the same
152 matched overlapping distance [9],[34]. The formulation in Figure.7 assumes that the given
153 genome G={TGGGACTGG}, then the genome G is sequenced, resulting in an array of reads
154 R={GACT, ACTGG, TGGGAC }. According to computer science, concatenating these reads
155 based on overlapping is correct if and only if the reads have an exact suffix-prefix match at a
156   specific overlapping distance. Thus, so both assembled genomes {GACTGGGAC} and
157   {TGGGACTGG} are correct solutions accordingly. However, because of the ambiguity that is
158 resulted from repeats in short reads, when the genome is assembled genome to be
159 {GACTGGGAC} while the real one is {TGGGACTGG}, this problem is a well-known problem
160 called misassembly [17].
161 Most de novo overlapping algorithms have a high degree of misassembly resulting from repeat
162 ambiguity. The greedy algorithm is one of the simplest algorithms in a data structure. It is used
163 for overlapping. It starts with joining the two reads that overlap the highest. Unassembled reads
164 are chosen for the next run until a pre-defined minimum threshold is reached [9]. Although the
165 greedy approach is computationally feasible, it cannot distinguish the repeats, merging reads

166 wrongly resulting in misassembly [9] as shown in the example in Figure.8 for the Genome
167 G={GGATGGGGATGCCT} that has two repeats "ATG". Overlap Layout Consensus (OLC) is
168 a famous graph-based algorithm introduced by Staden in 1980 and subsequently improved and
169 extended by others. Many *de novo* assembly tools are based on OLC methods, such as Newbler,
170 PCAP, Celera Celera, CAP3, and ARACHNE. In OLC, repeats in short reads create a
171 mathematical problem of finding a Hamiltonian path. The graph will be too complex if the
172 genome is highly repetitive, such as the human genome.
173 Another widely used algorithm is De Bruijn Graphs (DBG), which Nicholas Govert de Bruijn
174 introduced in the 1940s. DBG considers each read is a k-mer from the Genome [9]. The first step
175 in DBG is to divide each read with length k into two substrings (edges) with length (k-1). Next,
176 draw each left substring corresponding to its right substring. A path that contains every edge of
177 the graph exactly once is called the Eulerian path or Eulerian walk. DBG is a widely used
178 algorithm. Current de novo assemblers use DBG, such as Euler, Velvet, SOA Pdenovo, ABySS,
179 and IDBA. However, repeats in short reads create a fragmented assembly problem in this
180 algorithm, shown in Figure.9.
181 In the reference-guided approach, repetitive sequences can result in ambiguity during the
182 alignment process when a read has approximately matches with many positions (offsets) in the
183 reference genome. This ambiguity is a well-known data structure problem called the multiple-
184 matching problem. However, multiple-matching results in misassembly, as shown in Figure.10.
185 Most alignment tools cannot accurately map reads against the reference genome. The sensitivity
186 or confidence of the Alignment tool (S) is calculated by the ratio of correctly mapped read to
187 incorrectly mapped read at a particular threshold (S=Number of reads mapped correctly/a number
188 of reads mapped incorrectly). Repeats in short reads affect the sensitivity of the alignment tool,
189 resulting in a degree of misassembly, decreasing the total accuracy of assembly [18].
190

191 **Computational Performance Challenges in Genome Assembly**
192 The second computational problem in genome assembly is challenging performance due to the
193 massive data amount generated by high coverage NGS. This massive data amount generates big
194 performance challenges in each assembly approach in terms of CPU, memory, and storage.
195 Assembly performance challenges exist in both assembly approaches [14]. In the reference-
196 guided approach, reads alignment works in time complexity O(NL). N is the number of reads
197 (millions) to be aligned against each offset of reference genome with length L, which might be
198 billions of letters. It is a highly intensive process, as shown in Figure.11. On the other hand,
199 computational complexity in *de novo* overlapping works in time complexity O(N)^2, where (N) is
200 the number of reads (millions). Comparing millions of reads against millions to detect
201 overlapping is a well-known data structure querying problem called all-against-all or all-suffix-
202 prefix problem (ASPP), as shown in Figure.11 [8].
203

204

205

Accuracy And Performance Evaluation in Genome Assembly

The accuracy of the genome assembly produced through the assembly approaches can be evaluated using metrics scores throughout the assembly process, such as counting, the number of contigs, the proportion of reads that can be aligned against the known reference genome if exists (misassembly degree), and the absolute length of contigs. A commonly used statistical metric is N50. Given a set of contigs, each with its own length, the N50 is defined as the shortest contigs' length in the group of contigs which represent 50% of the assembly length. N50 can be used to denote the contiguity of the assembly. The pseudocode of how to calculate N50 from an array of contigs is shown in the pseudocode in Box.1 [48]. Although N50 is still widely used for assembly evaluation, this metric does not reflect the quality of an assembly and can be misleading [21]. An example of using N50 to evaluate the genome assembly is a *Ciona intestinalis* genome with an estimated N50 of 234 kb length , which was found inaccurate a few years later due to repeat problems that resulted in misassembly. The N50 metric does not consider that some contigs may be erroneously joined or even overlapped [58] [62]. U50 is similar to N50 in the calculation, and the only difference is that U50 uses unique contigs for calculation that do not overlap with other contigs to represent a more accurate result [21]. Another method to ensure the completeness of an assembly is by detecting the presence-absence variations (PAV) against the reference genome. It identifies the sequences in the reference genome that are entirely missing in the newly generated assembly [58]. Also, QUAST-LG is a tool that compares large genomic de novo assemblies against reference sequences and calculates the quality metrics. [59]. The Merqury, is a reference-free tool assembly evaluation By comparing k-mers in a *de novo* assembly to those found in unassembled high-accuracy reads [60][61].

The performance of genome assembly can also be evaluated through the computational execution time (time complexity), referred to as time complexity, which is represented as O(N), where (N) is the size of the input and (O) is the execution time for the algorithm, it can be linear, logarithmic, quadratic, exponential or any other relationship. For example, if the number of reads to search for overlapping is five reads, and the index hits ten times, the time complexity of overlapping is O(2N), as shown in the pseudocode in Box.2. Also, the computational performance of the genome assembly can also be assessed by memory or storage consumption.

Survey methodology

This review article summarizes the enhancement in genome assembly accuracy using prior repeat identification and hybrid assembly approaches. Also, we highlight the limitations of current methods. On the other hand, we overview the most famous data structure indexing to optimize genome assembly, such as k-mer index, tree index, hash index, also parallelization methods. We also highlight the limitations and show that combining more than one method improves accuracy and performance. This review helps computer science researchers understand genome data and its considerations, such as repeat. Also, it will help to identify the main research problems in the genome assembly area in terms of accuracy and performance optimization.

249   **I.**   **Search Strategy**

250   The authors found the primary studies through seven digital libraries: Web of Science, Science
251   Direct, SCOPUS, Google Scholar, IEEE Explore, PubMed, and EMBASE. These digital libraries
252   are most preferred and widely used by the research community. The search query string is
253   applied to an individual database to find relevant studies based on the research questions shown
254   below in box.3. The search strategy for the literature process is shown in Figure.12. A total of 65
255   research papers, after the duplications were removed, were found and downloaded from the
256   mentioned database.

257   **II.**   **Inclusion criteria**

258   The studies concerning that are:
259   A. Written in English
260   B. Based on genome assembly.
261   C. Focused on accuracy and performance optimization.
262   D. Published in conference or journal
263
264
265   **III.**   **Current Solutions for Misassembly Problem**
266
267   **A. Prior Repeat Identification Methods**
268   Repeat identification is a method to detect and identify repetitive sequences in the reads or in the
269   reference genome [1]. Identifying repetitive sequences before genome assembly helps the
270   assembly algorithm distinguish the reads that have repeats, and avoid misassembly, thus,
271   significantly enhancing the accuracy of genome assembly from short reads [23]. Repeat
272   identification methods are introduced first in the reference-guided approach through the repeat
273   masker method. However, it did not work well with big genomes [1]. The de novo approach is
274   preferable, identifying repeats from target genome reads rather than a reference genome [6].
275   Most of the current repeat identification methods are statistically based, calculating the
276   occurrences of repetitive sequences in reads. In order to clearly understand how it works, let (G)
277   is the genome that is sequenced with 5X coverage, so (C=5). Let rs is a repetitive sequence with
278   a length of 9 letters, rs={ACGTGATAT}. Let R denote the array of reads generated by
279   sequencing genome (G), R={r1,r2…ri} as shown in Figure.13. Reads are just a substring of
280   Genome (G). According to the current sequencing coverage, we expect any unique substring of
281   (G) to appear in (R) several times less than or equal to (C). In this example, a read, ri{GTG}
282   appears in the data set six times, which means the frequency of ri, fri=6, while the sequencing
283   coverage C =5. If ri sequence is a unique sequence in Genome (G) the max fri =5 under coverage
284   (C=5) and max fri =6 under coverage (C=6), and so on. If (frs=1) in (G), it is impossible that the
285   frequency of the entire rs or even any substring of rs exceeds the sequencing coverage (C) in the
286   reads dataset. In Figure. 13, fri =6, greater than C, simply means that the extra occurrence of ri
287   comes from sequencing another copy of rs. If the read frequency is less than or equal to the
288   sequencing coverage, it means that this read comes from a unique sequence in the genome. On
289   the other hand, if the frequency of the read exceeds the well-known sequencing coverage, it

290   means that this read comes from the repetitive sequence in the genome. More than half of some
291   genomes are repeats and vary in length, which might be too long, such as TE Thus the size of repetitive
292   sequences is used as a criterion to evaluate the accuracy of identifying repeats in many repeat
293   identification methods [3][63][64]. Many algorithms have been proposed according to the previous
294   statistical concept. Earlier, the RePS algorithm assumed that any 20-mer (substring from the read
295   with length 20 letters) that appears in the dataset more often than a sequencing coverage is likely
296   to be an exact repeat and is masked out. A sliding window assembly (SWA) is a genome
297   assembly algorithm proposed by [24] for identifying and assembling repeats. Fundamentally, the
298   algorithm considers the entire read is a substring of the genome. SWA calculates the frequency
299   of each read in the dataset. Based on the well-known sequencing coverage SWA splits reads into
300   two groups, repeat and non-repeat, assembling them separately. The non-repeat group consists of
301   those reads with a frequency less than the sequencing coverage, while the repeat group includes
302   those with a frequency greater than the sequencing coverage. Then, the overlap is run for each
303   group separately in parallel to construct the genome superstring. However, SWA considers the
304   entire reads as the substring of genome superstring, which means it can detect repeat length that is only
305   greater than or equal to the read length, while repetitive sequences might be smaller than the read's length,
306   such as microsatellite tandem repeat, which cannot be detected by SWA [1] as shown in Figure.14.
307   Repeat *De novo* (REPdenovo) is a repeat identification method proposed by [25], similar to the
308   earlier method RepARK that splits reads into smaller substrings (k-mers) then identifies the
309   frequency of k-mers, group them to repeat and non-repeat groups, finally, assembles these k-
310   mers. Unlike SWA, REPdenovo uses the average k-mers frequency as a threshold of this
311   algorithm, where a repeat is identified when a k-mer frequency is more than the threshold, as
312   shown in Figure.15. REPdenovo is also able to identify a repeat shorter than the read length by
313   constructing them from reads' k-mers to longer repeats. Evaluation and comparison show that
314   REPdenovo outperforms the earlier RepARK method regarding the accuracy and completeness
315   of repeats construction. REPdenovo discovered that previous repeat annotations had missed a
316   significant number of 190 potentially new repeats in the human genome. The high accuracy of
317   identifying the repeat enhances the genome's contiguity with N50 3141, while RepARK method only N50
318   116. However, when reads are chopped up into k-mers the biological info represented in the
319   reads is lost [12]. Similar to REPdenovo, the Detection of Long Repeats (DLR) converts all
320   reads into unique k-mers of a certain length and screens out the k-mers with a high frequency
321   [26]. The Detection method in DACCOR assembly is also similar to REPdenovo calculation
322   introduced as a stage in Characterization Reconstruction (DACCOR) hybrid assembly. Unlike
323   REPdenovo, it splits the reference genome into k-mers instead of the reads themselves, then
324   identifies repeats from k-mers of the reference genome, similarly to REPdenovo calculation.
325   The detection method in DACCOR assembly identifies repeats from the reference genome,
326   which might come with low identification accuracy, simply because repeats in the reference
327   genome might be slightly different from the real repeat in target genome reads [1].
328

329   On the other hand, digital signal processing can also detect repetitive elements. In most signal-
330   processing methods, DNA sequences are converted to numerical sequences, and repetitive

331 elements can be identified by Fourier power [49]. However, capturing the essential features of
332 repetitive elements, such as copy numbers of repeats, is still challenging. In addition, Fourier
333 transform cannot capture repeats in short genomes [49].
334 Recently, machine learning has been employed for identifying repetitive elements. Repeat
335 Detector (Red) has been proposed by [27] as de novo tool for discovering repetitive elements in
336 genome reads. Red utilizes a Hidden Markov model (HMM) dependent on labeled training data,
337 and it successfully identified new repeats in the Human Genome. However, it only generates
338 genome coordinates for repeats without any annotation. Therefore, the red output does not help
339 analyze repeat content or TEs evolution[1]. Improvements are required in optimizing
340 computation requirements and choosing suitable training datasets for such a machine-learning
341 program in order to strengthen the program accuracy and precision on the repeat prediction in the
342 genomes [28].
343 As shown in the current limitations of some current repeat identification methods in Table.1. No
344 one algorithm fits all lengths of repetitive sequences [29]. The combination of two or more
345 methods might introduce a better solution to identify and assemble repeat accurately. A
346 combination of repeat identification methods consistently outperforms the usage of a single
347 method due to the variety of repetitive sequences. An example of this combination is merging
348 consensus sequences generated by RepeatModeler and Repbase library, which successfully
349 annotated many repeats in many genomes [1]. Another example of a repeat identification
350 methods combination is Tandem Repeats Finder (TRFi), used to identify tandem repeats and
351 combined with the RepeatExplorer method, and the result outperforms both of them in
352 identifying complex tandem repetitive sequences [30]. However, combining methods adds more
353 computational complexity, which might be addressed by data indexing and increasing the
354 parallelism level during the identification process.
355

356 **B. Hybrid Assembly Approach**
357 The last two decades have seen significant improvement in solving the misassembly problem
358 that resulted from repeats through the hybridization concept. Fundamentally, hybridization
359 combines two or more methods to perform genome assembly. Hybridization appears in genome
360 assembly in different ways. It can combine data from different platforms [44] or two or more
361 genome assembly methods. The example is combining the OLC and the DBG method to
362 introduce DBG2OLC to enhance assembly accuracy that is affected by repeats [13]. The
363 hybridization between de novo and reference-guided approaches creates a hybrid assembly
364 hybrid approach which able to complement each approach's limitation [32]. The idea of a hybrid
365 assembly approach has been introduced by [33], who proposed a scaffold-builder assembler
366 composing an initial de novo assembly from reads , followed by aligning contigs against a
367 reference genome to build a scaffold. Enhancing assembly accuracy through the hybrid assembly
368 approach adds many contributions to understand the diversity of species.. An example of is a
369 study on Cyclospora cayetanensis species, , a parasite that caused intestinal infection, has a
370 highly repetitive genome that could not be assembled accurately using de novo alone. Combining
371 a reference-guided approach with a de novo approach generates new assembly for this species
372 with a significantly greater depth of coverage and a lower degree of misassembly [7].

373   Although, accurate reconstruction of repetitive sequences cannot be without repeat resolution
374   (repeat identification). Most of the current assembly hybrid approaches do not employ prior
375   repeat identification [23]. The accuracy is enhanced by using more than one reference genomes
376   or multiple samples of the target genome from different sequencing platforms [7]. The
377   characterization and reconstruction of repetitive regions (DACCOR) introduced a new idea to
378   enhance the hybrid assembly approach based on identifying repetitive sequences in the reference
379   genome prior to hybrid assembly. Although the detection repeat identification method in
380   DACCOR identifies repetitive sequences from the reference genome, the results on the bacterial
381   genome *Treponema pallidum,* as shown in Table.2, reveals high accuracy in identifying
382   repetitive sequences [23]. In Table.3, the comparison of the accuracy in generating contigs from
383   repetitive regions detecting repeats for T. pallidum (sample AR1 with the coverage X157),
384   DACCOR successfully constructed 96.8% of contigs that can be mapped to the reference
385   genome, while, SPAdes de novo assembly generates about 69.5% of contigs from repetitive
386   regions that can be mapped to the reference genome and EAGER reference guided approach with
387   generates around 42.1%. Using repetitive sequences from the reference genome to detect repeats
388   in the target genome in DACCOR might have advantages over the conventional assembly
389   approach. However, the repeats identified through the approach might be limited to the reference
390   genome, which may not well represent the actual repetitive sequences present in the target
391   genome. On the other hand, the hybrid assembly approach comes with big performance
392   challenges. The total assembly time in the hybrid approach is exceptionally challenging for any
393   traditional algorithm or hardware architecture. Although performance challenges are well-
394   optimized in each approach individually [36], no data structure indexing is introduced to fit
395   overlapping and reads alignment together in the hybrid assembly approach.
396
397   **IV.   Current Solution For Performance Optimization Of Genome Data**
398   **A. Indexing in Genome assembly**
399   Indexing could speed up the search by reducing the number of iterations (time complexity),
400   leading the algorithm to focus on a specific part of the data instead of searching in the entire
401   dataset [50]. One of the earliest data structure indexing methods is k-mer index. However,
402   massive genome data requires more optimization for this index. The keys of k-mer index can be
403   extracted as a shape, skipping letter instead of extracting all k-mers from the string to speed up
404   the index hitting. As shown in Figure.16, k-mers are extracted as a pattern by taking the first and
405   third letters followed by the fifth and sixth letters. This reduces the index total size, which
406   improves the specificity of the index hitting and index verification. This kind of index must be
407   hit with the same index pattern. This idea was implemented in Homopolymer Compressed k-
408   mers method proposed by [36]. This implementation added significant enhancement of mapping
409   reads to the reference genome. Another variation to speed up querying k-mer index is a binary
410   search. Binary search, also known as half-interval search, logarithmic search is a search
411   algorithm that finds the position of a target value within a sorted array. The k-mers of the
412   genome must be ordered alphabetically in ascending order, as shown in Figure.17. In this case,
413   the query does not look up in the entire index simply because (TGG) is alphabetically greater
414   than (GTG). The query does bisection (dividing the problem into two halves), skipping the first
415   part of the index [31]. Each iteration bisection occurs, resulting in a significant reduce the query
416   time. The total number of bisections that are needed to perform can be calculated as $Log_2(n)$.

417   This implementation significantly reduces search trials during reads mapping to reference
418   genome and can be implemented for reads during the search for overlapping [51].

419

420   The Hash index is the most famous implementation of k-mer index. In the hash index, the hash
421   function is responsible for mapping each distinct k-mer to one bucket (group), as shown in
422   Figure.18. the Hash index is widely used in many studies on genome data for indexing dataset
423   reads [10] to reduce suffix-prefix search in de novo overlapping. However, because of repeat,
424   sometimes two keys may generate an identical hash causing both keys to point to the same
425   bucket, which is known as a hash collision, slowing down the search in the hash index [50].

426

427   Suffix indexes are another data structure index family. There are four variations of suffix
428   indexes: trie suffix tree, suffix array, and FM index. However, the suffix tree (ST) is the most
429   popular and widely used indexing tool in bioinformatics applications [16]. It is the base of some
430   popular sequence alignment tools, such MUMmer and RePuter, and can be used to solve more
431   complex problems such as repeat identification. Unfortunately, the construction of ST is highly
432   memory and CPU-consuming [52]. However, there are many research efforts to reduce the
433   construction of ST time complexity, for example, scaling the construction of ST on multiple
434   CPU cores (64 cores) that [37] achieves a speedup from 2X to 4X over the original algorithm to
435   construct S.T. Figure.19 shows the construction of ST index from genome G={ACGCGT}.

436

437   In overlapping massive genome reads, maybe performing an approximate match between suffix
438   and prefix first add a significant reduction of time complexity of overlapping by skipping the
439   exact match for suffix prefix candidates that do not have an approximate match. The pigeonhole
440   principle (PP) is a well-known data structure principle called seed and extends principle that can
441   achieve this idea. PP separates the read into non-overlapping partitions, and the exact matching
442   algorithm checks the match of one of the partitions considered suffix prefix approximate match,
443   then the verification step must be done as shown in Figure.20. According to this powerful
444   principle, many algorithms and methods for indexing introduced to genome data took advantage
445   of the pigeonhole principle and suffix-tree index. The result of *Escherichia Coli* genome shows
446   that the pigeonhole solution with prefix tree is superior in terms of time and storage compared to
447   the traditional suffix tree for overlapping [8], as shown in Table.4.

448

449   The optimization using data structure indexes for overlapping reads and read alignment in the
450   genome hybrid assembly approaches remains a challenge. However, ST is the best solution for
451   repetitive sequences scenarios, its construction memory, and CPU consumption. In Hash, index
452   collisions slow down the search at this key. FM.- index is based on Burrows-Wheeler
453   Transformation (BWT) that was proposed by Burrows and Wheeler in 1994 [38]. Querying FM
454   index cannot use binary search because it does not consist of the complete rotation of the genome
455   and needs another process to find the offset of the index hit, which is a highly computational task
456   in massive genome data.

457   Hybridization between two indexes might introduce a better solution for indexing massive
458   genome data. An example of the hybrid between two indexes is MiniSR index which uses k-mer
459   indexing with the hash index. MiniSR can index the massive human genome in a few minutes
460   [53]. The hybridization between a few indexed might introduce a solution to optimize
461   overlapping and read alignment in hybrid genome assembly.

462

463   **B. Parallelization in Genome Assembly**
464   The exponential growth of genomic data has instigated a significant challenge for genomics
465   analysis computing infrastructure and software algorithms. Genomic experiments are now
466   reaching the size of Terabytes and Petabytes [22]. Scientists may require weeks or months to
467   process this massive amount of data using their own workstations [22]. Parallelism techniques
468   and high-performance computing (HPC) environments can help to reduce the total processing
469   time [20], [22]. The simplest definition of parallelization is to split the extensive process into
470   smaller subprocesses and run them in parallel on multiple CPUs [52][22]. It solved many
471   performance challenges in the big data area. However, parallelization is introduced in genome
472   data for genome assembly in some stages individually.

473

474   Sequencing technologies produce millions to billions of short reads. The first step to assembling
475   these reads is to extract them from FASTA or FASTQ files into structured database tables, which
476   is too time-consuming. Parallelization could reduce raw data streaming time is to split big
477   FASTA or FASTAQ files into smaller files processed in parallel [56].

478

479   Another variation of parallelization in genome data is parallelizing the index's construction. An
480   example of this idea is proposed by [39] for speeding up prefix tree index construction by letting
481   each processor work on strings that start with a specific character in the alphabet (A, C, G, and
482   T). For example, processor 1 constructs the part of the tree that corresponds to strings that start
483   with the letter "A" while processors 2, 3, and 4 construct the parts of the tree corresponding to
484   the strings starting with "C", "G", and "T" respectively. A similar idea was introduced by [40],
485   proposing HipMer assembler to construct k-mer index using a deterministic function to map each
486   k-mer to a target processor.

487

488   In de nove overlapping and read alignment of reference-guided approach, the IBD algorithm
489   addressed pairwise comparison problems that can be used to overlap two reads or align reads
490   against k-mer of the reference genome [19]. The algorithm breaks the problem (N^2) using
491   massive parallelization with each order (N) comparison. This approach is applied to the U.K.
492   Biobank dataset, with a 250X faster time and 750X less memory usage over the standard
493   approach of pairwise alignment.

494

495

496   Over the last decades, many computing platforms parallelize different stages of genome
497   assembly. For instance, Hadoop[22] introduced a powerful idea of parallelizing the suffix tree
498   index construction. However, the Apache spark platform works 100 times faster than Hadoop,
499   especially in iterative operators [41]. A distributed and parallel computing tool named HAlign-II
500   was introduced by [22] to address read alignment with extremely high memory efficiency.
501
502
503   GraphSeq method proposed splitting genome files and using a high scaling platform [42].
504   GraphSeq works on the Apache spark platform and achieved 13X speedup *de novo* genome
505   assembly. GraphSeq splits a big compressed file into several small ones loading all of the reads
506   in parallel, generating the corresponding suffixes, grouping those suffixes with the exact initial
507   string into the same partition, and applying string graph construction in parallel by partitions.
508   Similarly, an innovative algorithmic approach proposed by [2] is called Scalable Overlap-graph
509   Reduction Algorithms (SORA), performing string graph reduction on Apache Spark. SORA was
510   evaluated with human genome samples to process a nearly one billion edge of string graph in a
511   short time frame with linear scaling.
512
513   Parallelization enhances the performance of streaming massive genome data, genome index
514   construction, and overlapping in the *de novo* approach or read alignment in the reference-guided
515   approach. Despite its performance, due to the nature of the hybrid assembly approach, to our best
516   knowledge, the usage of parallelization in overlapping and reads alignment for genome assembly
517   is yet to be fully implemented.
518
519   .
520

Conclusions

522
523   Assembling genome reads with the presence of repetitive sequences at a good quality is
524   essentially important but, indeed, far more challenging. Ignoring repeats in the genome would
525   revoke the purpose of the genome assembly. A massive number of reads produced by sequencing
526   technologies have caused high computational performance for the genome assembly approach. A
527   lot of research has contributed significantly to the development of enhancing assembly accuracy
528   by reducing the degree of misassembly in the genome assembly approaches. In this paper, we
529   suggest by using prior repeat identification methods to create prior repeat resolution guiding
530   overlapping algorithm in *de novo* or alignment algorithm in the reference-guided approach could
531   reduce the chance of genome misassembly. However, none of the repeat identification methods
532   can accurately detect all types with different lengths of repetitive sequences. The combination of
533   *de novo* with the reference-guided assembly approach could yield better results for genome
534   assembly, which can be another alternative solution. Although the hybrid assembly approaches
535   found to outperform the individual *de novo* approach and the reference-guided approach,
536   repetitive sequences can still generate misassembly, which could be enhanced through the use of

537 multiple samples or multiple reference genomes. However, using multiple samples or reference
538 genomes will add high computational complexity to the hybrid assembly approach. Moreover,
539 most of the current assembly hybrid approaches still lack repeat identification prior to the
540 approach, which can help to increase the accuracy of the genome assembled.
541 The combination between the *de novo* approach and the reference-guide approach comes with a
542 more computationally extensive performance challenge than using the approach individually.
543 Therefore, the investigation of optimizing the computational performance challenges in the
544 hybrid assembly approach for overlapping and reads alignment is still an open challenge.
545 Enhancement of the repeat identification accuracy by approaches would be one of the main
546 priorities in addressing issues of genome misassembly. The enhanced repeat identification
547 method should be added at the prior stage for hybrid genome assembly to enhance the accuracy
548 without relying on multiple samples or reference genomes, which can increase the computational
549 complexity of the the hybrid assembly approach.
550 Consequently, the performance of the hybrid assembly approach might be optimized through the
551 hybridization of indexing methods with parallelization to optimize overlapping and reads
552 alignment in the hybrid genome assembly approach.
553 Hybrid genome assembly enhanced by accurate prior repeat identification and optimized by a
554 combination of indexing and parallelization, as shown in Figure.21, might be a novel solution for
555 assembling repetitive genomes from short reads.

556
557

558 **Conflicts of Interest**
559 The authors declare no conflict of interest.
560

561 **ACKNOWLEDGMENT:**
562 This research was supported by the Ministry of Higher Education (MOHE) through Fundamental
563 Research Grant Scheme FRGS/1/2019/ICT02/UTM/02/13, Research anagement Centre (RMC),
564 UTM, and ALI@S research group.

565

566 **References**
567

568 1. Zeng, L., Kortschak, R.D., Raison, J.M., Bertozzi, T. and Adelson, D.L., 2018. Superior ab
569 initio identification, annotation and characterisation of T.E.s and segmental duplications from
570 genome assemblies. PloS one, 13(3), p.e0193588.
571 2. Paulson, H., 2018. Repeat expansion diseases. Handbook of clinical neurology, 147, pp.105-
572 123.
573 3. Genovese LM, Geraci F, Corrado L, Mangano E, D'Aurizio R, Bordoni R, Servergnini M,
574 Manzini G, Bellis GDe, D'Alfonso S, Pellegrini M. 2018 "A census of Tandemly repeated
575 polymorphic loci in genic regions through the comparative integration of human genome

assemblies," Frontiers in Genetics, 9, p.155. Available at: https://doi.org/10.3389/fgene.2018.00155.

4. Pinto, B.S., Saxena, T., Oliveira, R., Méndez-Gómez, H.R., Cleary, J.D., Denes, L.T., McConnell, O., Arboleda, J., Xia, G., Swanson, M.S. and Wang, E.T., 2017. Impeding transcription of expanded microsatellite repeats by deactivated Cas9. Molecular cell, 68(3), pp.479-490.

5. Patil, N., Toshniwal, D. and Garg, K., 2013. Genome data classification based on fuzzy matching. CSI Transactions on ICT, 1(1), pp.9-28.

6. Baichoo, S. and Ouzounis, C.A., 2017. Computational complexity of algorithms for sequence comparison, short-read assembly and genome alignment. Biosystems, 156, pp.72-85.

7. Gopinath, G.R., Cinar, H.N., Murphy, H.R., Durigan, M., Almeria, M., Tall, B.D. and DaSilva, A.J., 2018. A hybrid reference-guided de novo assembly approach for generating Cyclospora mitochondrion genomes. Gut Pathogens, 10(1), pp.1-8. Available at: https://doi.org/10.1186/s13099-018-0242-0

8. Haj Rachid, M. (2017) "Two efficient techniques to find approximate overlaps between sequences," BioMed Research International, 2017, pp. 1–8. Available at: https://doi.org/10.1155/2017/2731385.

9. Simpson, J.T. and Pop, M., 2015. The theory and practice of genome sequence assembly. Annual review of genomics and human genetics, 16, pp.153-172.

10. Zhang, H., Chan, Y., Fan, K., Schmidt, B., & Liu, W. (2018). Fast and efficient short read mapping based on a succinct hash index. BMC bioinformatics, 19(1), 1-14.

11. Garibyan, L. and Avashia, N., 2013. Research techniques made simple: polymerase chain reaction (PCR). The Journal of investigative dermatology, 133(3), p.e6.

12. Angeleska, A., Kleessen, S. and Nikoloski, Z., 2014. The Sequence Reconstruction Problem. In Discrete and Topological Models in Molecular Biology (pp. 23-43). Springer, Berlin, Heidelberg.

13. Jain, M., Olsen, H.E., Turner, D.J., Stoddart, D., Bulazel, K.V., Paten, B., Haussler, D., Willard, H.F., Akeson, M. and Miga, K.H., 2018. Linear assembly of a human centromere on the Y chromosome. Nature biotechnology, 36(4), pp.321-323.Available at: http://dx.doi.org/10.1038/nbt.4109

14. Jain, M., Olsen, H.E., Turner, D.J., Stoddart, D., Bulazel, K.V., Paten, B., Haussler, D., Willard, H.F., Akeson, M. and Miga, K.H., 2018. Linear assembly of a human centromere on the Y chromosome. Nature biotechnology, 36(4), pp.321-323.Available at: http://dx.doi.org/10.1038/nbt.4109

15. Lohmann, K. and Klein, C., 2014. Next generation sequencing and the future of genetic diagnosis. Neurotherapeutics, 11(4), pp.699-707.

16. Barsky, M., Stege, U., Thomo, A., & Upton, C. (2009, November). Suffix trees for very large genomic sequences. In Proceedings of the 18th ACM conference on information and knowledge management (pp. 1417-1420).

17. Medvedev, P., 2019. Modeling biological problems in computer science: a case study in 5genome assembly. Briefings in bioinformatics, 20(4), pp.1376-1383.

617   18. Thankaswamy-Kosalai, S., Sen, P. and Nookaew, I., 2017. Evaluation and assessment of
618   read-mapping by multiple next-generation sequencing aligners based on genome-wide
619   characteristics. Genomics, 109(3), pp.186-191.
620   19. Sapin, E., & Keller, M. C. (2021). Novel approach for parallelizing pairwise comparison
621   problems as applied to detecting segments identical by decent in whole-genome
622   data. Bioinformatics, 37(15), 2121-2125.
623   20. Ocaña, K. and de Oliveira, D., 2015. Parallel computing in genomic research: advances and
624   applications. Advances and applications in bioinformatics and chemistry, pp.23-35.Available at:
625   https://doi.org/10.2147/AABC.S64482
626   21. Lischer, H.E. and Shimizu, K.K., 2017. Reference-guided de novo assembly approach
627   improves genome reconstruction for related species. BMC bioinformatics, 18(1), p.474.
628   22. Shi, L. and Wang, Z., 2019. Computational strategies for scalable genomics analysis. Genes,
629   10(12), p.1017.Available at:https://doi.org/10.3390/genes10121017
630   23. Seitz, A., Hanssen, F. and Nieselt, K., 2018. DACCOR–Detection, characterization, and
631   reconstruction of repetitive regions in bacterial genomes. PeerJ, 6, p.e4742.
632   24. Lian, S., Li, Q., Dai, Z., Xiang, Q. and Dai, X.,. (2014) "Ade novogenome assembly
633   algorithm for repeats and nonrepeats," BioMed Research International, 2014, pp. 1–16.
634   Available at: https://doi.org/10.1155/2014/736473.
635   25. Chu, C., Nielsen, R. and Wu, Y., 2016. REPdenovo: inferring de novo repeat motifs from
636   short sequence reads. PloS one, 11(3), p.e0150719.
637   26. Liao, X., Zhang, X., Wu, F.X. and Wang, J., 2019, November. de novo repeat detection
638   based on the third generation sequencing reads. In 2019 IEEE International Conference on
639   Bioinformatics and Biomedicine (BIBM) (pp. 431-436). IEEE.
640   27. Girgis, H.Z., 2015. Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on
641   the genomic scale. BMC bioinformatics, 16(1), p.227.
642   28. Libbrecht, M.W. and Noble, W.S., 2015. Machine learning applications in genetics and
643   genomics. Nature Reviews Genetics, 16(6), pp.321-332.
644   29. Guo, R., Li, Y.R., He, S., Ou-Yang, L., Sun, Y. and Zhu, Z., 2018. RepLong: de novo repeat
645   identification using long read sequencing data. Bioinformatics, 34(7), pp.1099-1107.Available
646   at: https://doi.org/10.1093/bioinformatics/btx717
647   30. Peška, V., Sitová, Z., Fajkus, P. and Fajkus, J., 2017. BAL31-NGS approach for
648   identification of telomeres de novo in large genomes. Methods, 114, pp.16-27.
649   31. Berztiss, A. (2014) Data structures;: Theory and practice (Computer science and applied
650   mathematics). Ex-library. Academic Press. New York,USA.
651   32. Platt, R.N., Blanco-Berdugo, L. and Ray, D.A., 2016. Accurate transposable element
652   annotation is vital when analyzing new genome assemblies. Genome biology and evolution, 8(2),
653   pp.403-410.
654   33. Silva, G.G., Dutilh, B.E., Matthews, T.D., Elkins, K., Schmieder, R., Dinsdale, E.A. and
655   Edwards, R.A., 2013. Combining de novo and reference-guided assembly with scaffold_builder.
656   Source code for biology and medicine, 8(1), p.23.
657   34. Wang, P., Meng, F., Moore, B. M., & Shiu, S. H. (2021). Impact of short-read sequencing on the misassembly
658   of a plant genome. BMC genomics, 22(1), 1-18.

659   35. Acuña-Amador, L., Primot, A., Cadieu, E., Roulet, A., & Barloy-Hubler, F. (2018). Genomic
660   repeats, misassembly and reannotation: a case study with long-read resequencing of
661   Porphyromonas gingivalis reference strains. BMC genomics, 19(1), 1-24.
662   36. Liu, Y., Yu, Z., Dinger, M.E. and Li, J., 2018. Index suffix–prefix overlaps by (w, k)-
663   minimizer to generate long contigs for reads compression. Bioinformatics, 35(12), pp.2066-
664   2074.
665   37. Labeit, J., Shun, J. and Blelloch, G.E., 2017. Parallel lightweight wavelet tree, suffix array
666   and FM-index construction. Journal of Discrete Algorithms, 43, pp.2-17.
667   38. Wang, J., Wong, G.K.S., Ni, P., Han, Y., Huang, X., Zhang, J., Ye, C., Zhang, Y., Hu, J.,
668   Zhang, K. and Xu, X., 2002. RePS: a sequence assembler that masks exact repeats identified
669   from the shotgun data. Genome research, 12(5), pp.824-831.
670   39. Haj Rachid, M. and Malluhi, Q. (2015) "A practical and scalable tool to find overlaps
671   between sequences," BioMed Research International, 2015, pp. 1–12. Available at:
672   https://doi.org/10.1155/2015/905261.
673   40. Ellis, M., Georganas, E., Egan, R., Hofmeyr, S., Buluç, A., Cook, B., Oliker, L. and Yelick,
674   K., 2017, August. Performance Characterization of De Novo Genome Assembly on Leading
675   Parallel Systems. In European Conference on Parallel Processing (pp. 79-91). Springer, Cham.
676   41. Wan, S. and Zou, Q., 2017. HAlign-II: efficient ultra-large multiple sequence alignment and
677   phylogenetic tree reconstruction with distributed and parallel computing. Algorithms for
678   Molecular Biology, 12(1), pp.1-10.vailable at:
679   https://doi.org/10.1186/s13015-017-0116-x
680   42. Su, C.T., Chang, M.T., Cheng, Y.C., Li, Y.L. and Wang, Y.T., 2018. GraphSeq: Accelerating
681   String Graph Construction for De Novo Assembly on Spark. bioRxiv, p.321729." Available at:
682   https://doi.org/10.1101/321729.
683   43. Röhling, S., Linne, A., Schellhorn, J., Hosseini, M., Dencker, T., & Morgenstern, B. (2020).
684   The number of k-mer matches between two DNA sequences as a function of k and applications
685   to estimate phylogenetic distances. Plos one, 15(2), e0228070.
686   44. Chen, Z., Erickson, D. L., & Meng, J. (2020). Benchmarking hybrid assembly approaches for
687   genomic analyses of bacterial pathogens using Illumina and Oxford Nanopore sequencing. BMC
688   genomics, 21(1), 1-21.
689   45. Baxevanis, A.D., 2020. Biological sequence databases. Bioinformatics. 4th Edition ed. New
690   York: John Wiley & Sons, pp.1-18.
691   46. Kim, J., Ji, M., & Yi, G. (2020). A Review on Sequence Alignment Algorithms for Short
692   Reads Based on Next-Generation Sequencing. IEEE Access, 8, 189811-189822.
693   47. Venuto, D. and Bourque, G., 2018. Identifying co-opted transposable elements using
694   comparative epigenomics. Development, growth & differentiation, 60(1), pp.53-62.
695   48. Manchanda, N., Portwood, J.L., Woodhouse, M.R., Seetharam, A.S., Lawrence-Dill, C.J.,
696   Andorf, C.M. and Hufford, M.B., 2020. GenomeQC: a quality assessment tool for genome
697   assemblies and gene structure annotations. BMC genomics, 21(1), pp.1-9.
698   49. Yin, C., 2017. Encoding DNA sequences by integer chaos game representation. arXiv
699   preprint arXiv:1712.04546.

700   50. Xiaolei, W., Wubin, Q., Chenggang, Z. and Dongsheng, Z., 2015. Kmer-indexer: A Fast K-mer
701   Indexing Program. Studies in health technology and informatics, 216, pp.1083-1083

702   51. Brodsky, L., Kogan, S., BenJacob, E., & Nevo, E. (2010). A binary search approach to
703   whole-genome data analysis. Proceedings of the National Academy of Sciences, 107(39), 16893-
704   16898.

705   52. Pingali, K.D., Tanay, K.C.P. and Baruah, P.K., 2017, March. GPU accelerated suffix array
706   construction for large genome sequences. In Innovations in Information, Embedded and
707   Communication Systems (ICIIECS), 2017 International Conference on (pp. 1-6). IEEE.

708   53. Bayat, A., Deshpande, N.P., Wilkins, M.R. and Parameswaran, S., 2018. Fast Short Read De-
709   Novo Assembly Using Overlap-Layout-Consensus Approach. IEEE/ACM transactions on
710   computational biology and bioinformatics, 17(1), pp.334-338.

711   54. Ekblom, R. and Wolf, J.B., 2014. A field guide to whole-genome sequencing, assembly and
712   annotation. Evolutionary applications, 7(9), pp.1026-1042.

713   55. Kulkarni, P. and Frommolt, P., 2017. Challenges in the Setup of Large-scale Next-
714   Generation Sequencing Analysis Workflows. Computational and Structural Biotechnology
715   Journal, 15, pp.471-477.

716   56. Pan, T., Flick, P., Jain, C., Liu, Y. and Aluru, S., 2017. Kmerind: A flexible parallel library
717   for k-mer indexing of biological sequences on distributed memory systems. IEEE/ACM
718   transactions on computational biology and bioinformatics.

719   57. Shafin, K., Pesout, T., Lorig-Roach, R., Haukness, M., Olsen, H. E., Bosworth, C., ... &
720   Paten, B. (2020). Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly
721   of eleven human genomes. Nature biotechnology, 38(9), 1044-1053.

722   58. Giordano, F., Stammnitz, M. R., Murchison, E. P., & Ning, Z. (2018). scanPAV: a pipeline
723   for extracting presence–absence variations in genome pairs. Bioinformatics, 34(17), 3022-3024.

724   59. Mikheenko, A., Prjibelski, A., Saveliev, V., Antipov, D., & Gurevich, A. (2018). Versatile
725   genome assembly evaluation with QUAST-LG. Bioinformatics, 34(13), i142-i150.

726   60. Rhie, A., Walenz, B. P., Koren, S., & Phillippy, A. M. (2020). Merqury: reference-free
727   quality, completeness, and phasing assessment for genome assemblies. Genome biology, 21(1),
728   1-27.

729   61. Chen, Y., Zhang, Y., Wang, A. Y., Gao, M., & Chong, Z. (2021). Accurate long-read de
730   novo assembly evaluation with Inspector. Genome biology, 22(1), 1-21.

731   62. Castro, C. J., & Ng, T. F. F. (2017). U50: a new metric for measuring assembly output based
732   on non-overlapping, target-specific contigs. Journal of Computational Biology, 24(11), 1071-
733   1080.

734   63. Taylor, A.S., Barros, D., Gobet, N., Schuepbach, T., McAllister, B., Aeschbach, L., Randall,
735   E.L., Trofimenko, E., Heuchan, E.R., Barszcz, P. and Ciosi, M., 2022. Repeat Detector: versatile
736   sizing of expanded tandem repeats and identification of interrupted alleles from targeted DNA
737   sequencing. NAR Genomics and Bioinformatics, 4(4), .lqac089.Available at:
738   https://doi.org/10.1093/nargab/lqac089.

739   64. Liao, X., Li, M., Hu, K., Wu, F. X., Gao, X., & Wang, J. (2021). A sensitive repeat
740   identification framework based on short and long reads. Nucleic acids research, 49(17), e100-
741   e100.
742   65. Guiglielmoni, N., Houtain, A., Derzelle, A., Van Doninck, K., & Flot, J. F. (2021).
743   Overcoming uncollapsed haplotypes in long-read assemblies of non-model organisms. BMC
744   bioinformatics, 22(1), 1-2

Box 1(on next page)

Box 1 N50 Metric Calculation

Let C array of scaffolds lengths C={c1, c2, c3,….,ci}

Let AL is Sum C[]

Let AL50= Sum C[]/2

Let incL is the incremental length with initial value incL=0

Let CN50[] is an empty array to store N50 scaffolds' members

Let i is the length of array C[]

Sort C descending

For n=1 to n=i

Do

IF (incL= AL50)

Break For Loop

ELSE

incL= incL + C[n].Length

Add C[n] to CN50[]

End IF

End For

return CN50[]

Thus:

N50=min(CN50[])

Box 1   **N50 Metric Calculation**

Box 2(on next page)

Index hit represent the overlapping computational time complexity

> Let N=5 reads
> Let All-against-all $O(N)^2$ =25 times
> Current index hit =10
> Thus: Overlapping computational time complexity is O(2N) which is
> liner algorithm.

5   Box 2   **Index hit represent the overlapping computational time complexity**

Box 3(on next page)

Box 3 Search string

2
3

(Genome assembly OR genome analysis OR indexing genome data OR genome misassembly OR parallelize genome data OR repetitive genome sequence identification) AND (accuracy OR performance OR enhancement) AND (method OR approach) AND (De novo approach OR reference guided approach OR hybrid approach) AND language (English)

5   Box 3   **Search string**

Figure 1

Figure 1 Biological activities of TE and TR change genome size

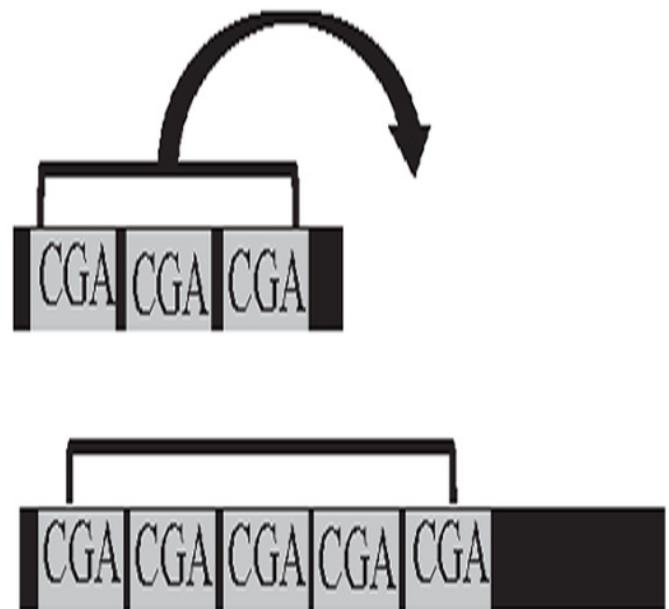



Figure 2

Figure 2 String exact and approximate match

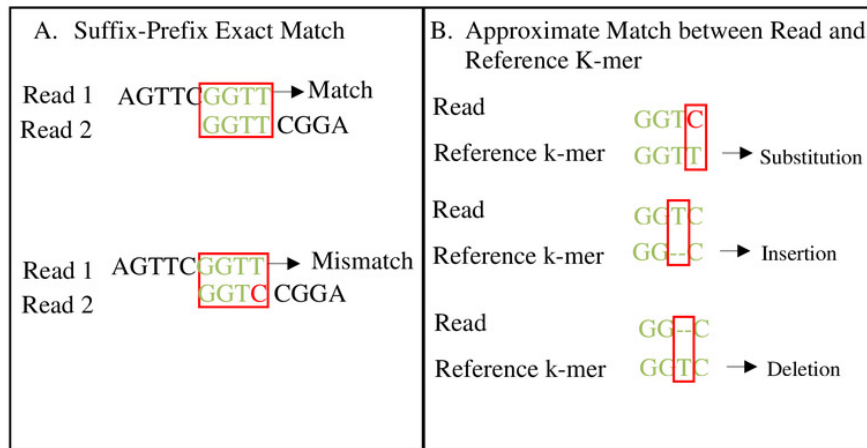



Figure 2   **String exact and approximate match**

Figure 3

Figure 3 Overlapping suffix-prefix match

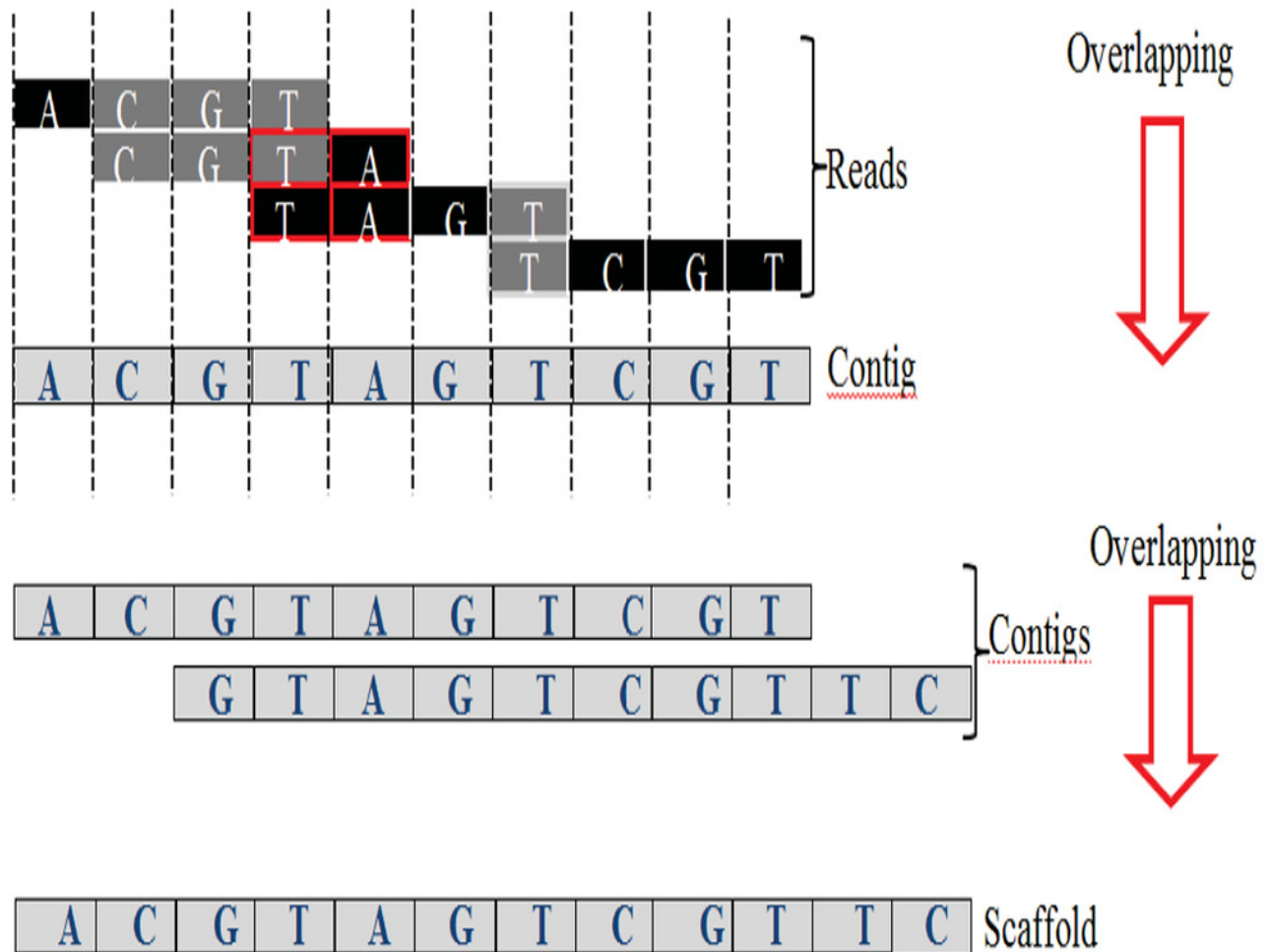



Figure 4

Figure 4 The impaction of overlapping length on assembly accuracy

Given Genome (G): ACTTCACGTCGTCGTCGTTGATCAA

Long Overlapping Length

Short Overlapping Distance

Figure 5

Figure 5 Reads mapping against reference genome (reads alignment)

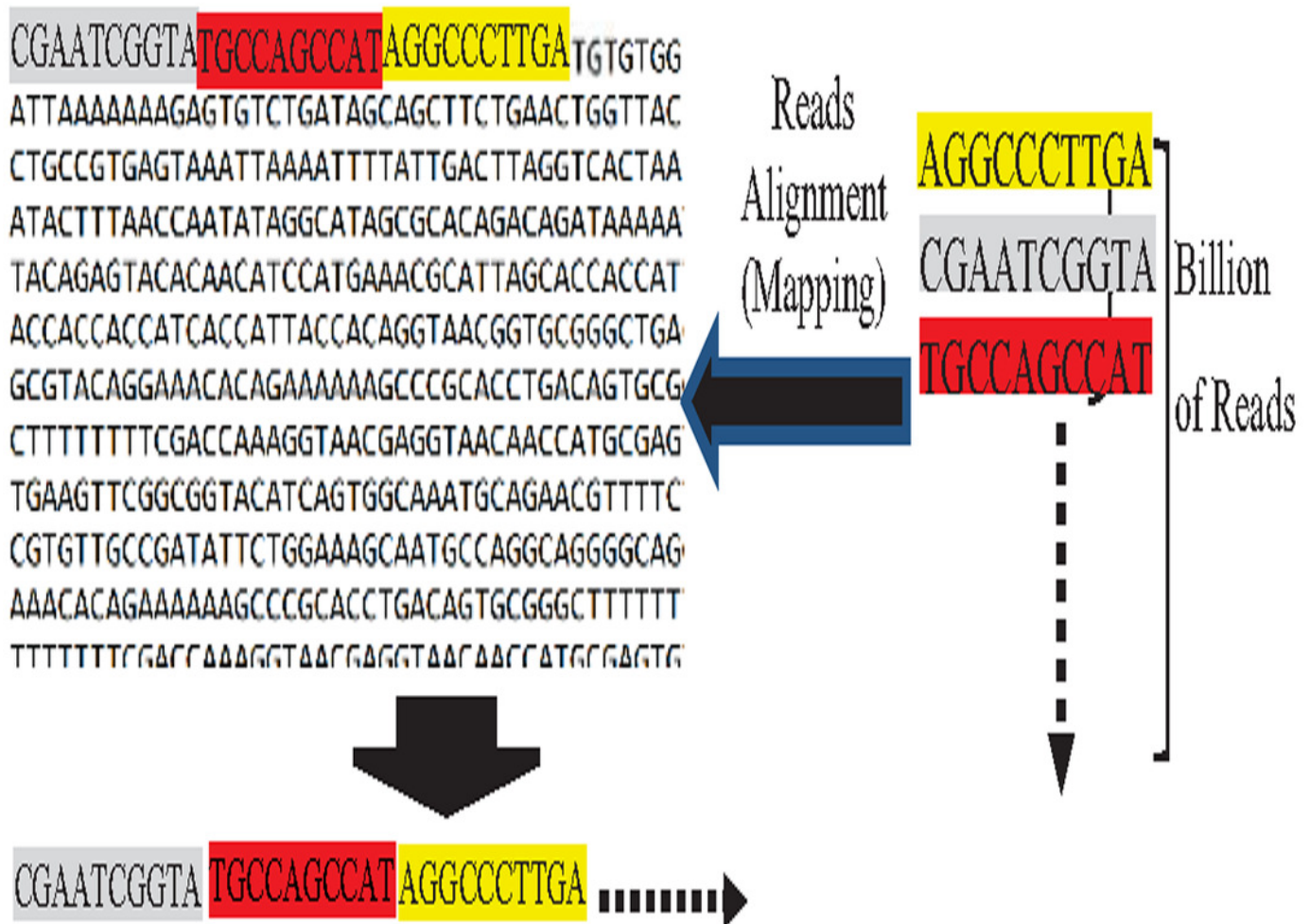



Figure 6

Figure 6 Sequencing coverage depth

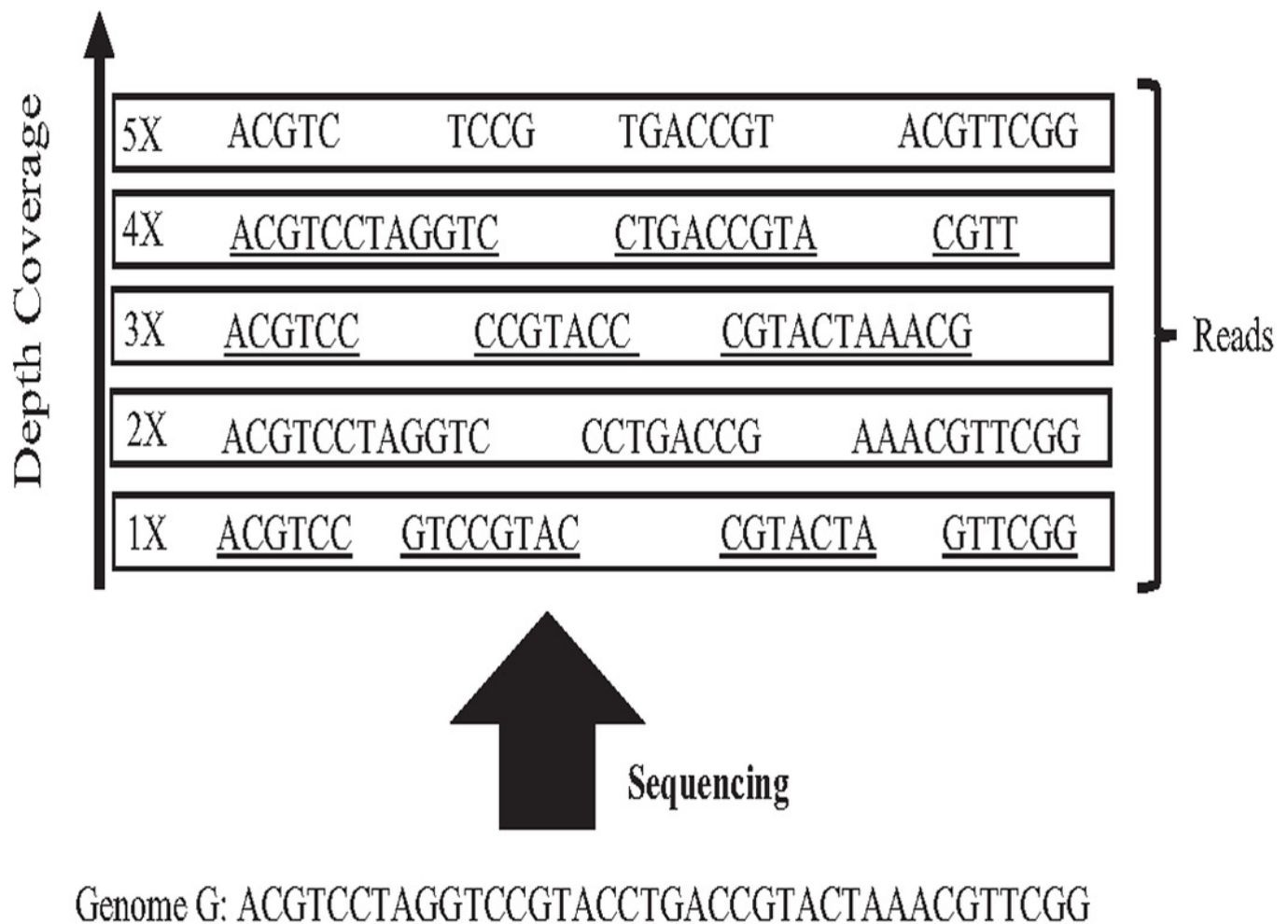



Figure 7

Figure 7 Mis-assembly in de novo approach

Figure 8

Figure 8 Mis-assembly in greedy algorithm

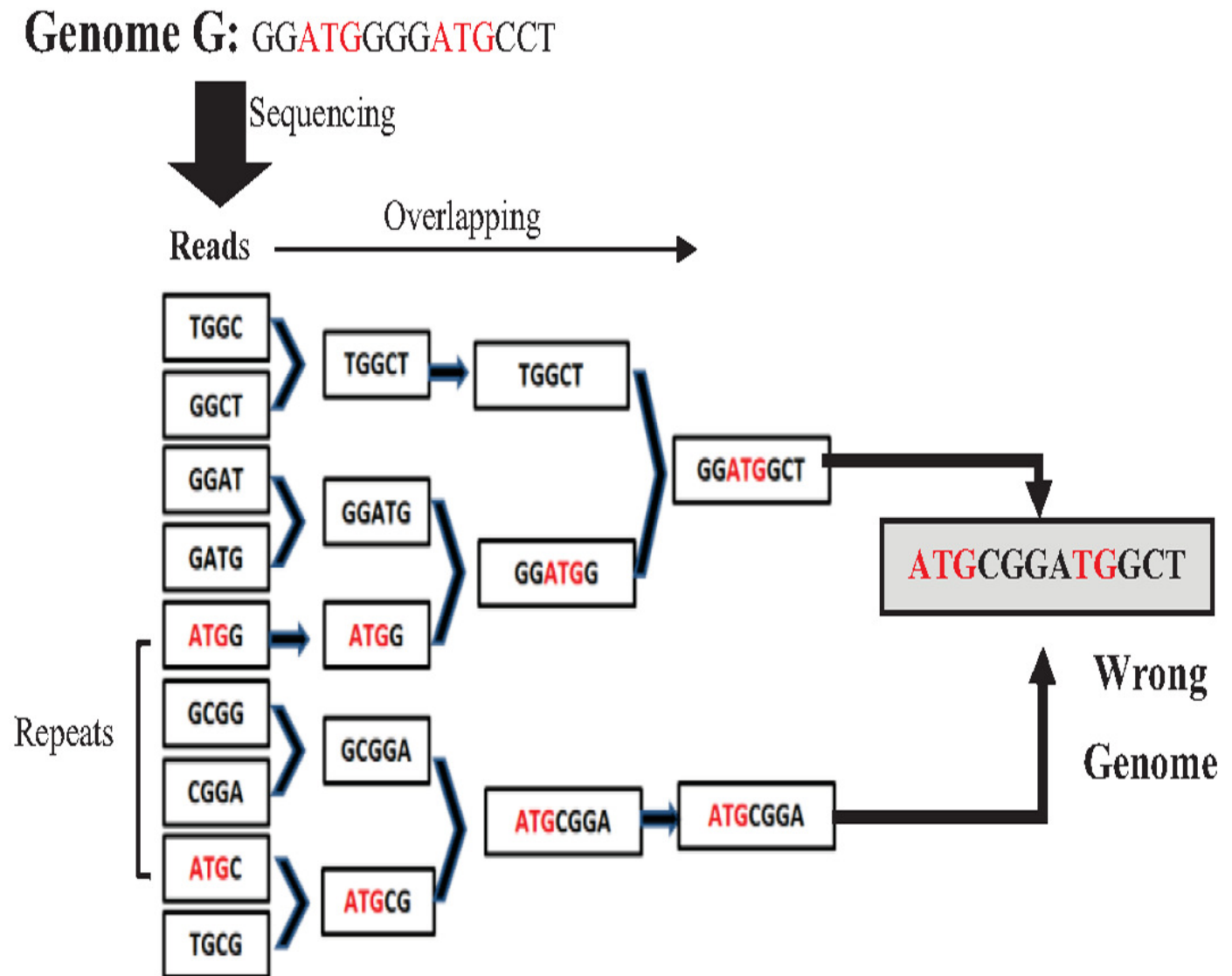



Figure 9

Figure 9 Fragmented assembly in DBG algorithm

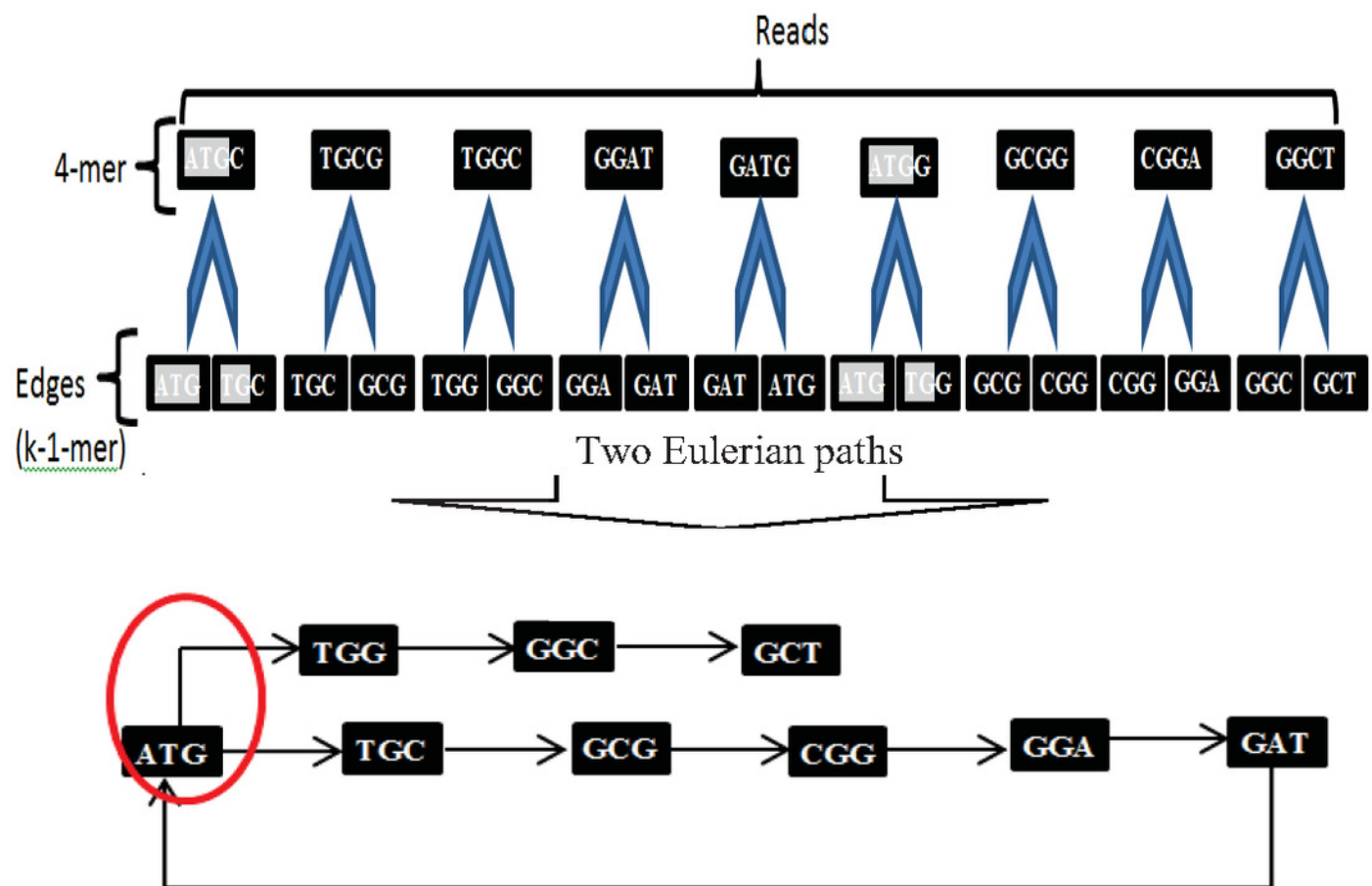



Figure 10

Figure 10 Mis-assembly in reference-guided approach

Input from biology to computer Science:

R={GGATGCGATGGCT} Reference Genome--------------Input 1

R={ATGCG, GAGTT, ATGCA } read to be aligned--------Input 2

Input String: ATGCG, GAGTT, ATGCA

Reads alignment

Reference Genome (G): ATGCGAGGCAGAGTT

Case 1 ATGCG ATGCAGAGTT *Correct Genome*

Case 2 ATGCA ATGCG GAGTT *Wrong Genome*

Mis-Assembly

Output from computer science to biology (if case 2):

ATGCAATGCGGAGTT which is wrong genome

Figure 11

Figure 11 Overlapping and reads alignment time complexity

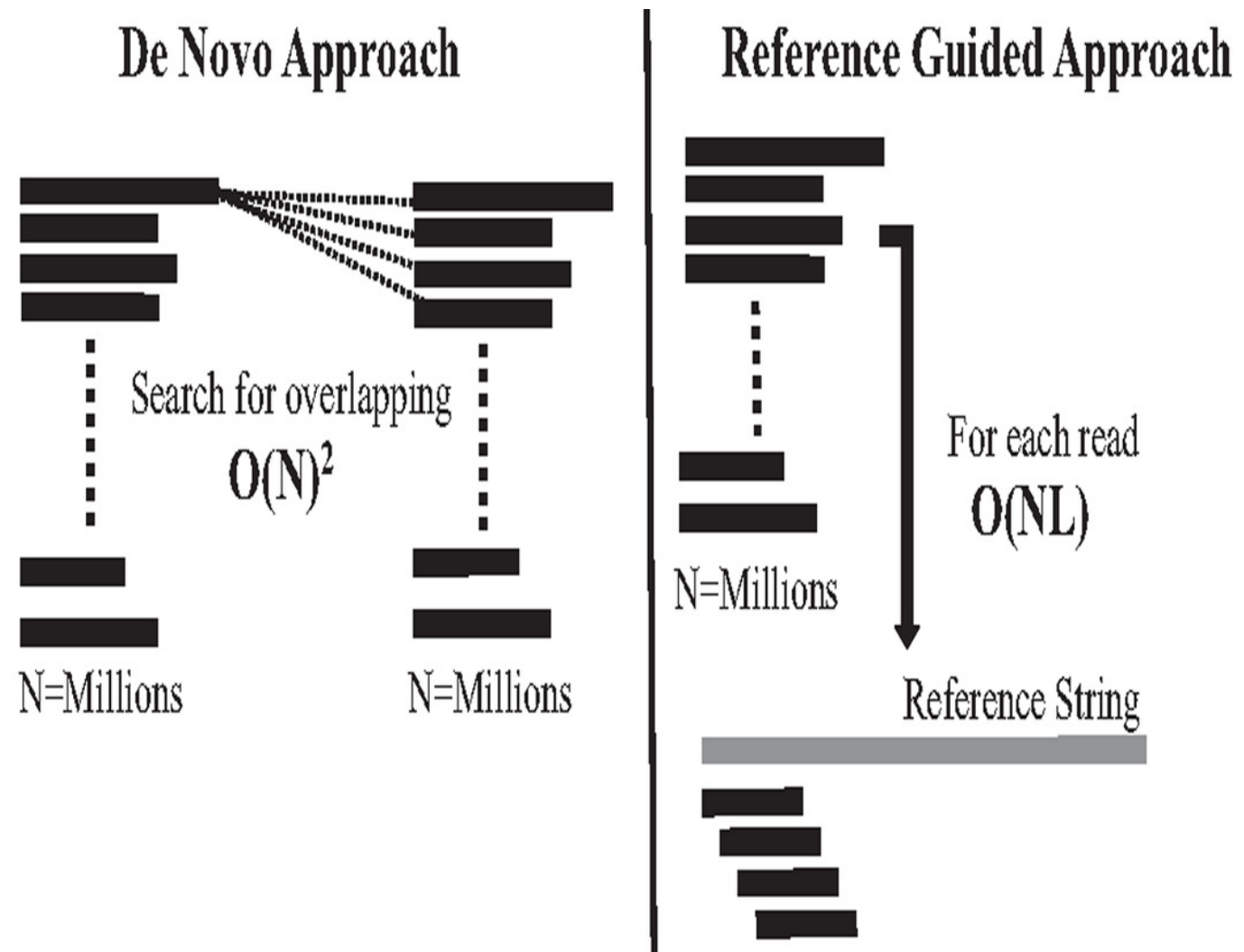



Figure 12

Figure 12 Search Map

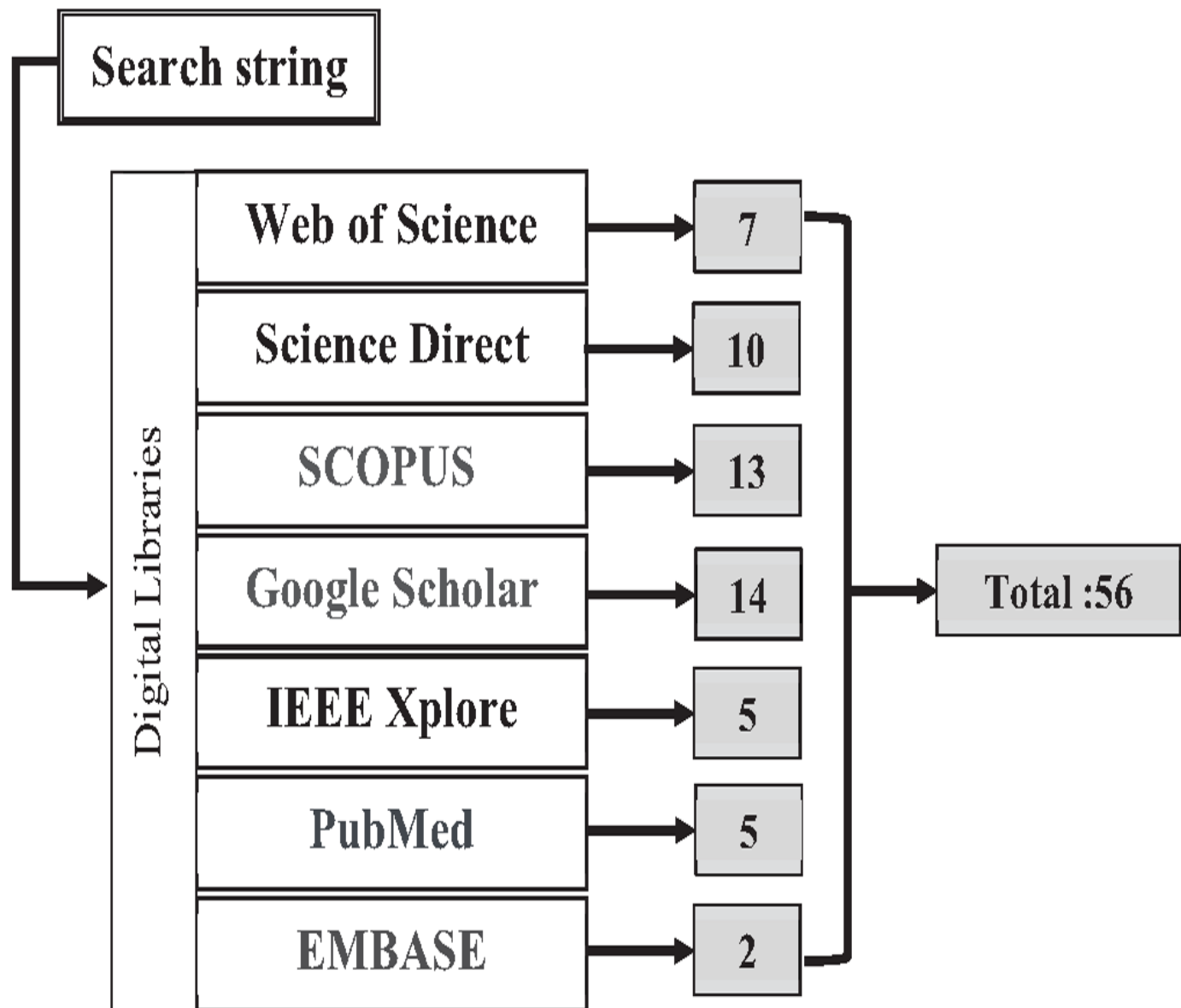



Figure 13

Figure 13 Repetitive sequences frequency greater than coverage depth

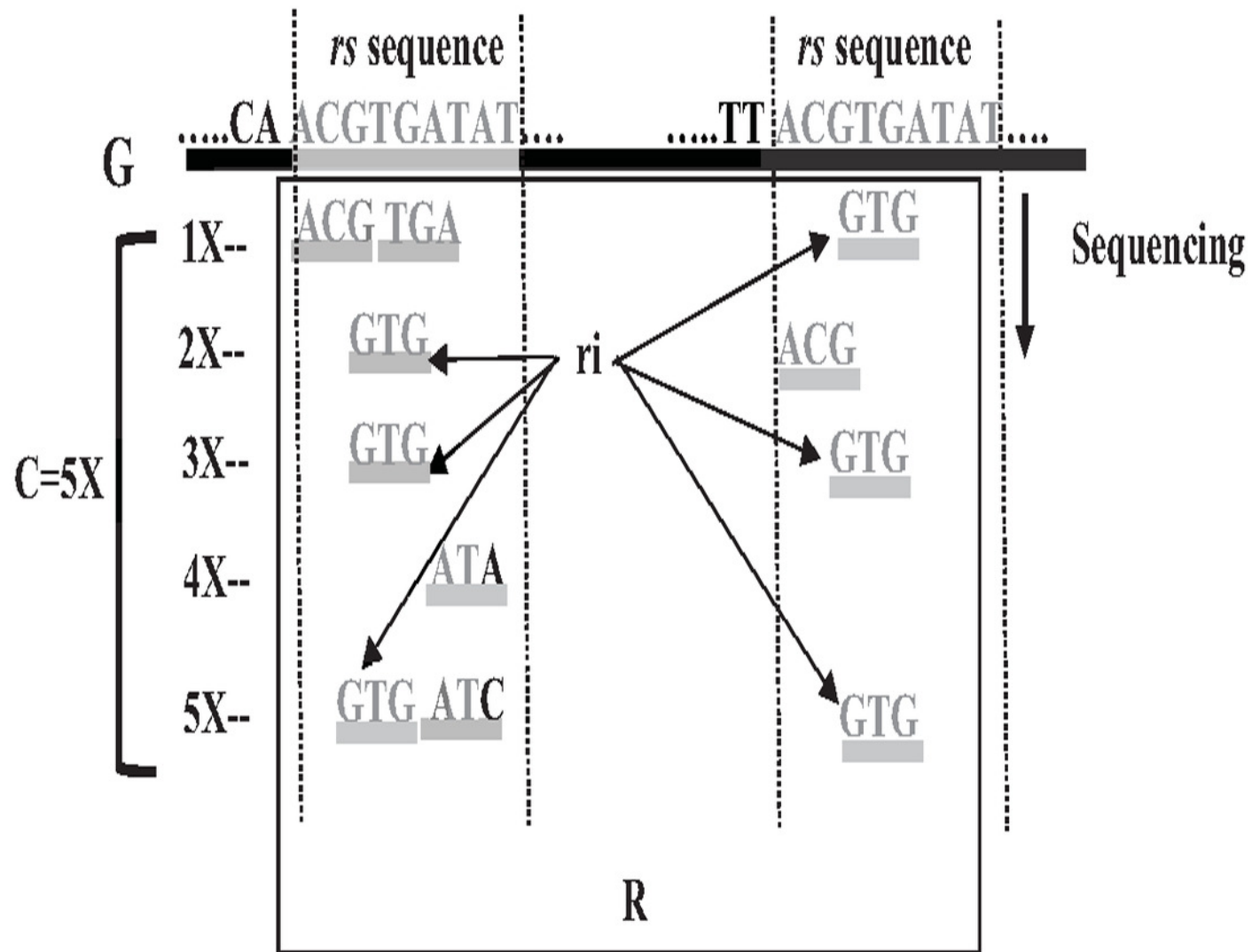



Figure 14

Figure 14 Repetitive sequence as substring of read

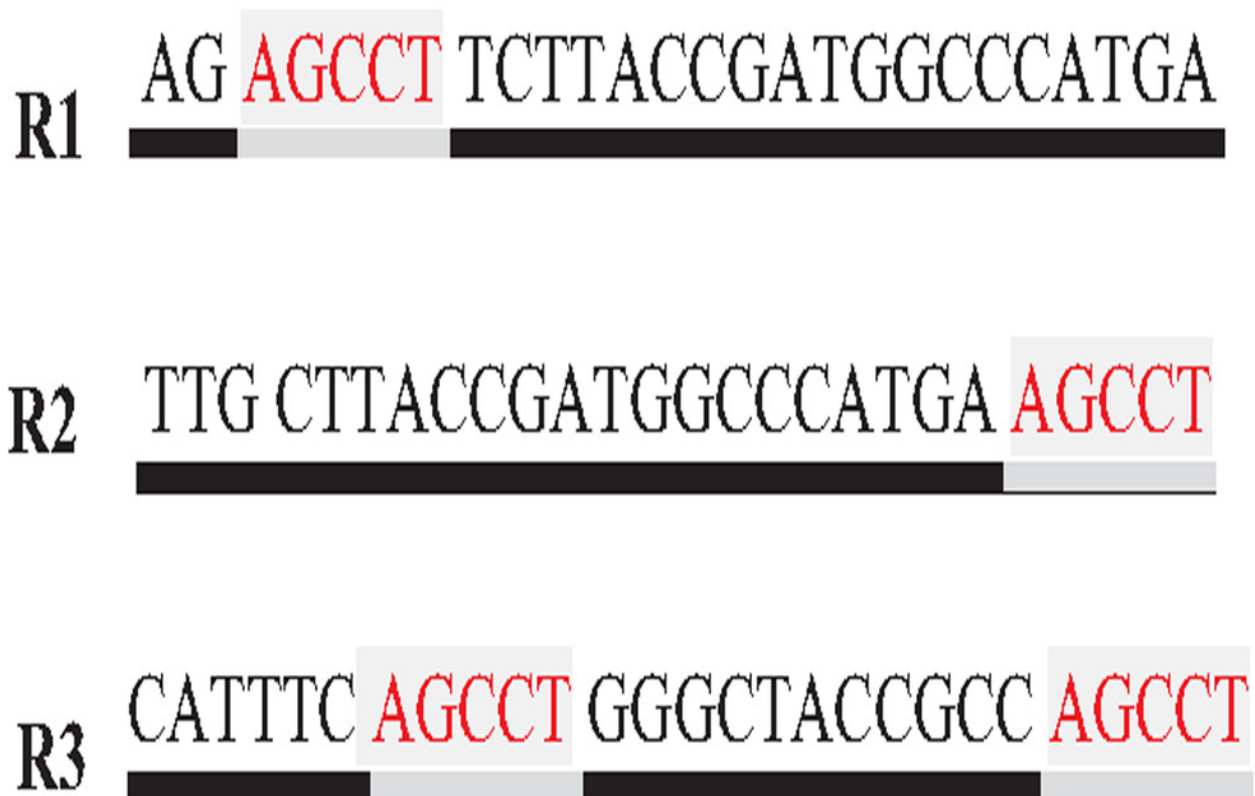



Figure 15

Figure 15 Reads' K-mers Frequency in REPdenovo

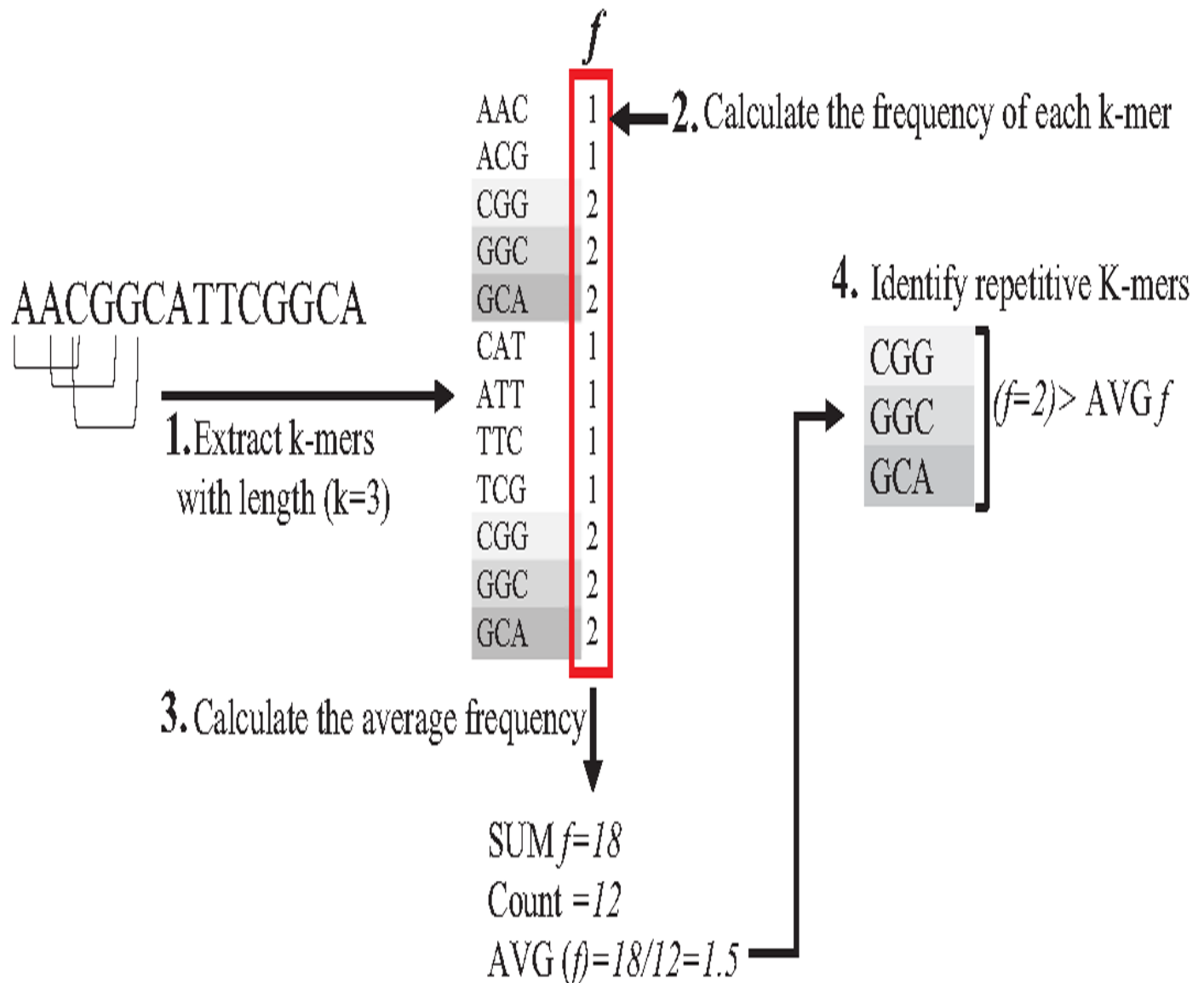
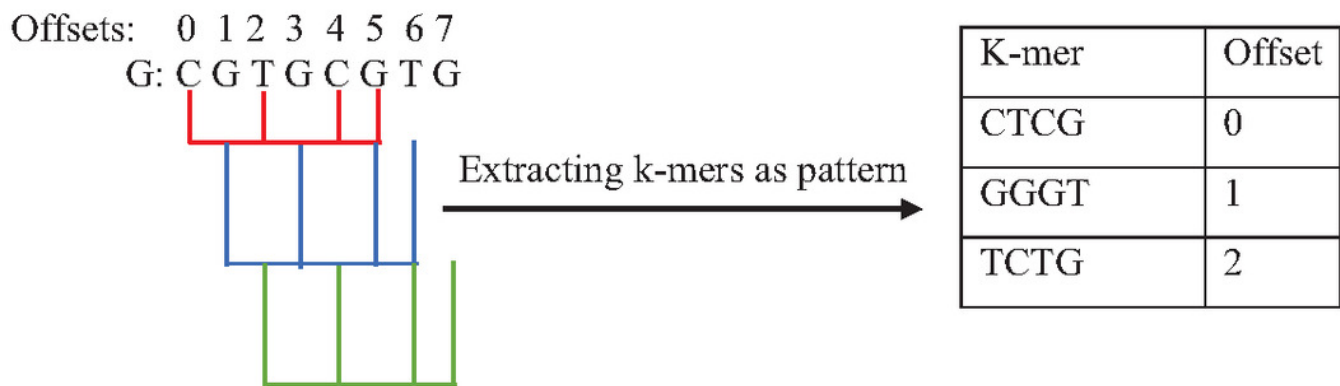



Figure 16

Figure 16 Querying k-mer index with pattern match

Query the previous index:

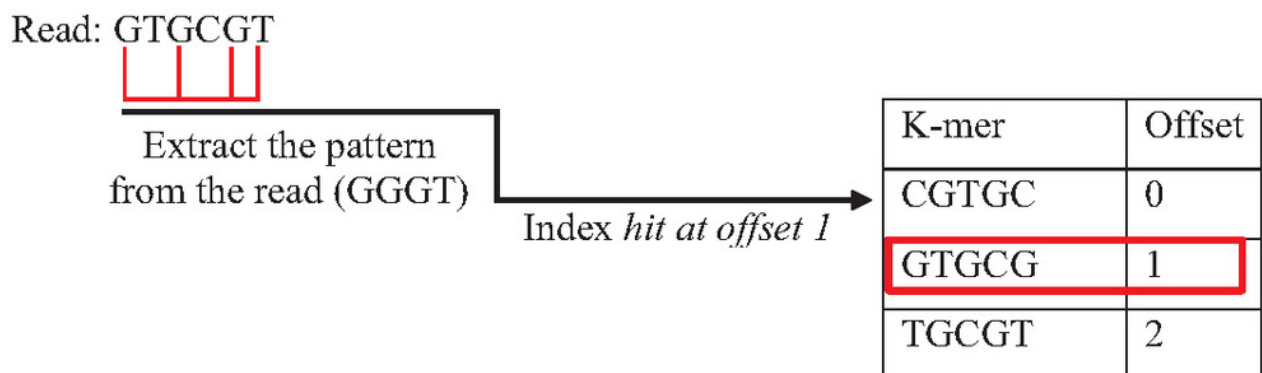



Figure 17 Binary search to querying k-mer index

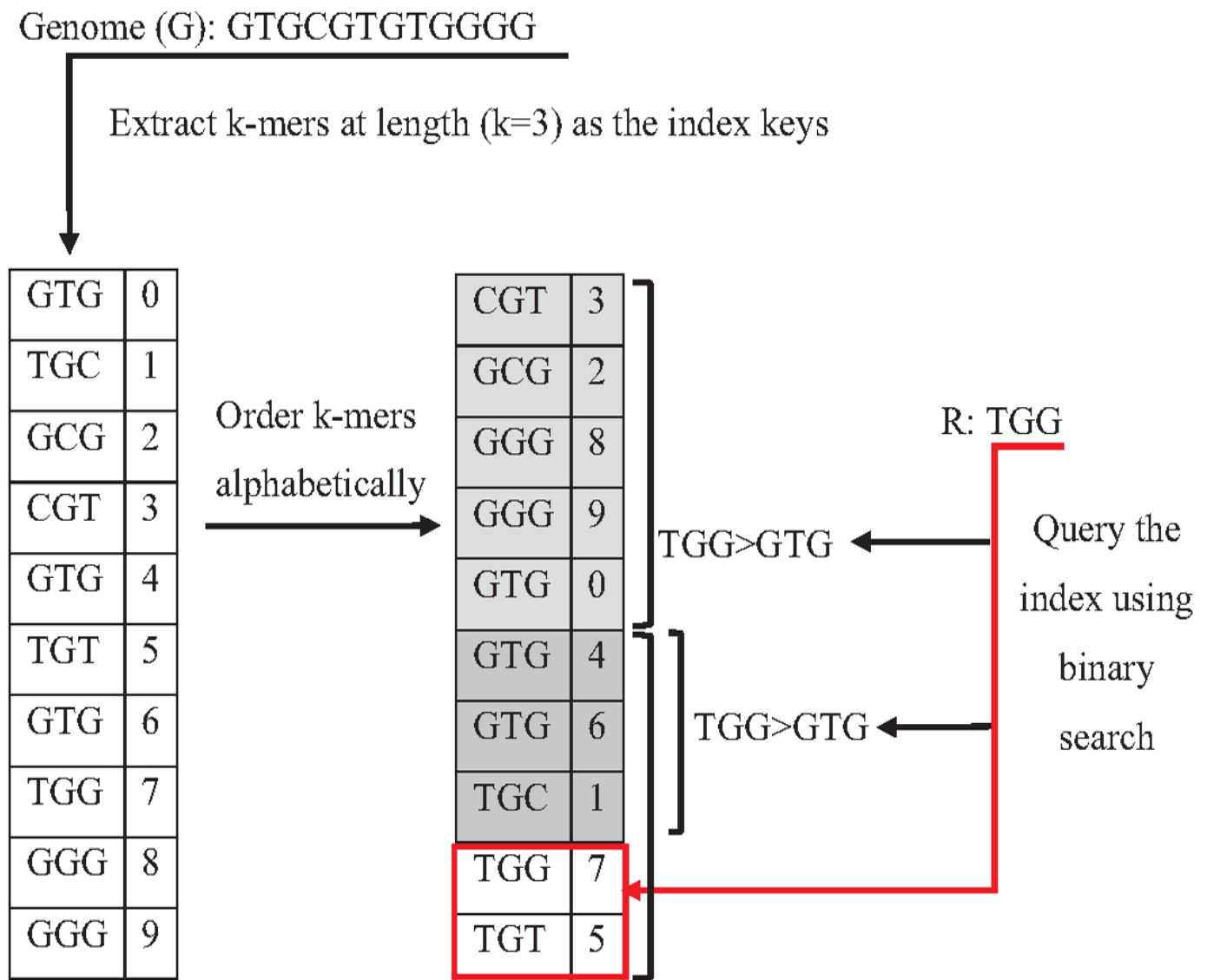



Figure 18

Figure 18 Construction of hash index and collision resulted by repeat

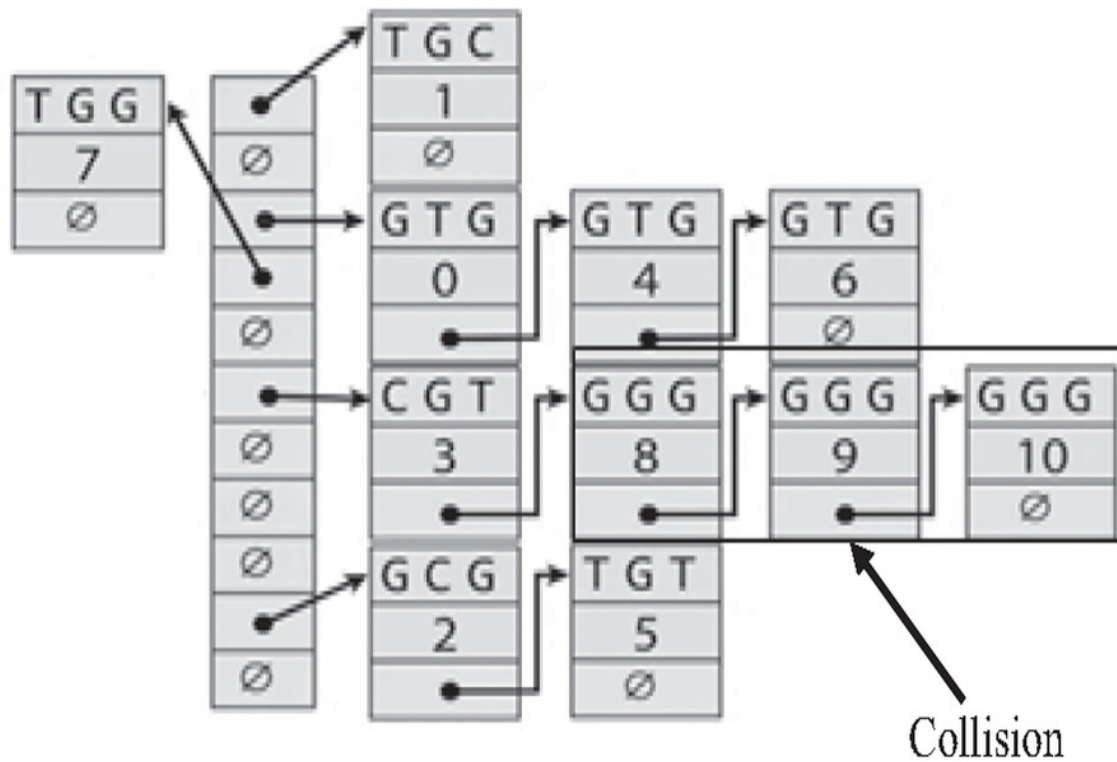



Figure 19

Figure 19 Prefix-tree index construction

Genome (G)={ACGCGT$}

Add (T$)

T$ at offset 5

Add (GT$)

Add (CGT$)

Add node GCGT$, There is a node GT$ that start with G also, splitting of node GT$ is required

Add (CGCGT$)

Add (ACGCGT$)

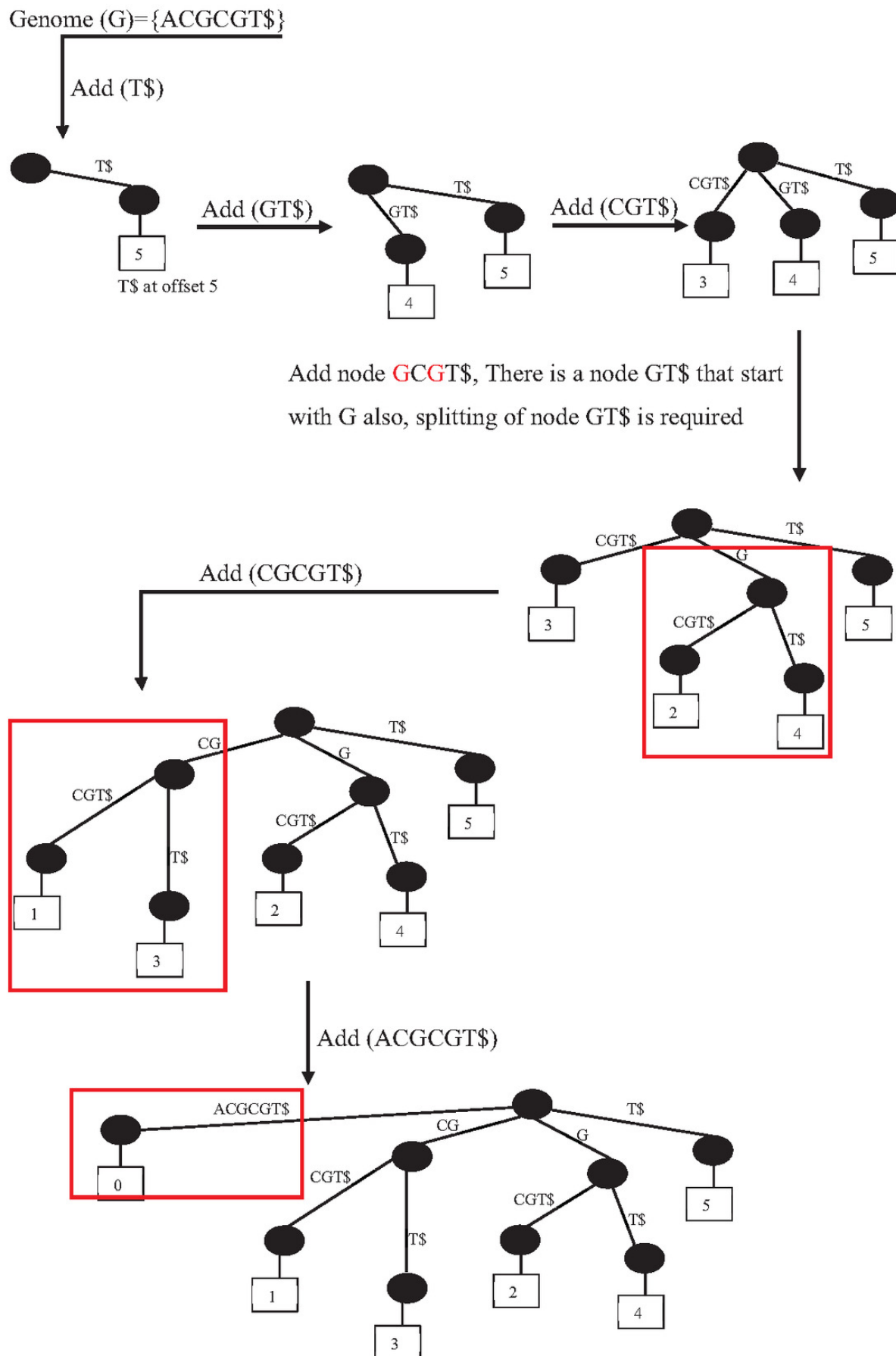



Figure 20

Figure 20 Pigeonhole principle in overlapping

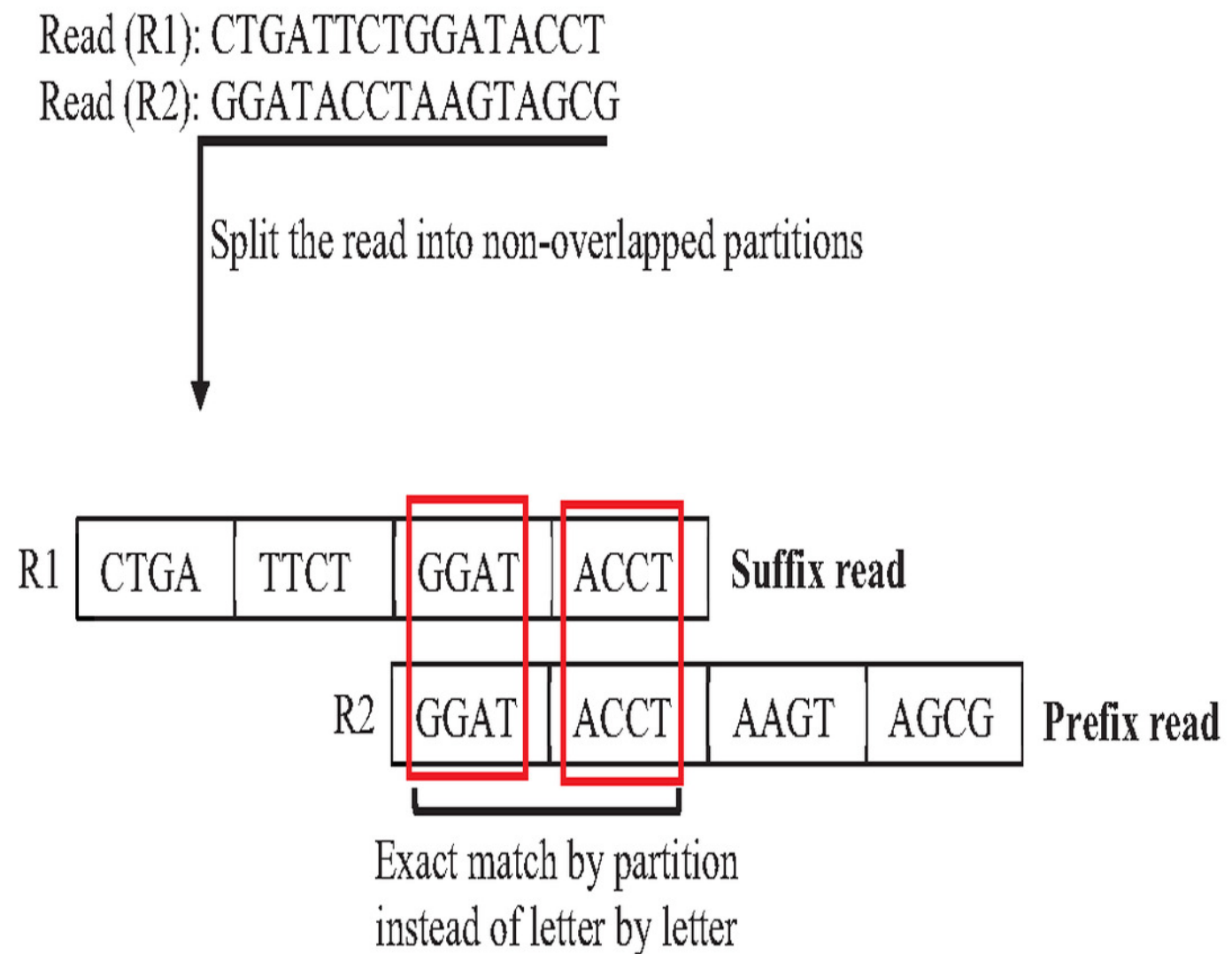



Figure 21

Figure 21 Solution to propose accurate and optimized assembly hybrid approach

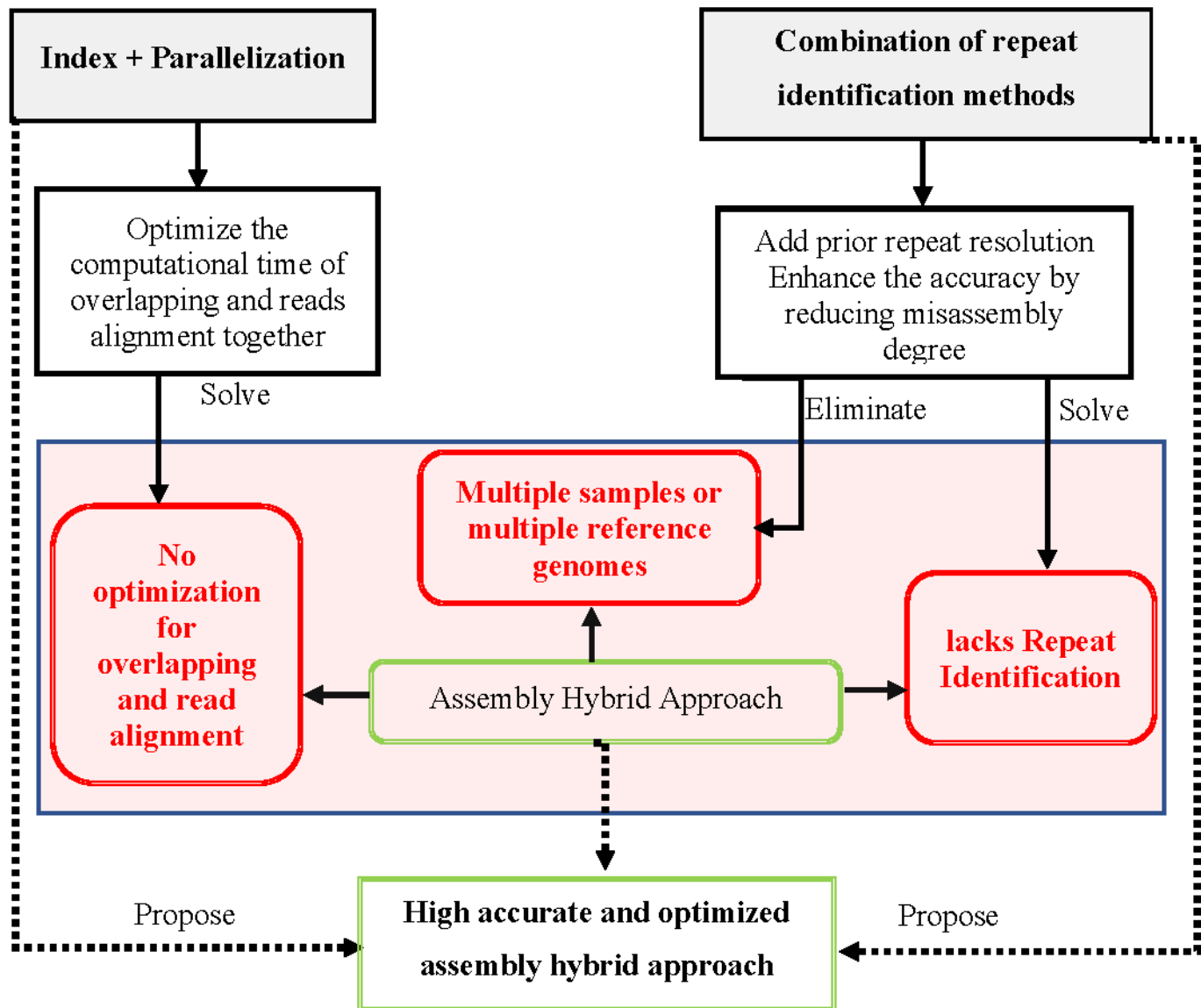



Table 1 The limitations of current repeat identification methods

Table 1 The limitations of current repeat identification methods

Repeat Identification Method	Limitations
SWA	Cannot detect repetitive sequence shorter than read's length
Detection in DACCOR assembly	Identify repetitive sequences from the reference genome that are slightly different from the target genome resulting in low identification accuracy.
REPdenovo	Biological info represented in read structure are lost when the reads are chopped up into k-mers
Digital signal processing	Difficult to capture a copy number of repeats and cannot capture repeats in short DNA sequences which means that many types of tandem repeats cannot be identified
Machine learning	Not useful for analysing repeat content such as transposon element for biological evolution and big performance challenge on massive genome data

Table 2 Repeat identification in Treponema pallidum Genome using Detection method in DACCOR

Table 2 Repeat identification in *Treponema pallidum Genome using Detection*

~~3~~ *method in   DACCOR*

Detected repetitive Sequences	*Treponema pallidum*
True positives	22382
True negatives	1116478
False positives	0
False negatives	773
Accuracy(%)	99.93

Table 3 DACCOR Enhance the genome resolution compared to SPAdes and EAGER method

2   **Table 3**   **DACCOR Enhance the genome resolution compared to SPAdes and EAGER**

3   **method**

Sample	Method	*Contigs from repetitive regions mapped against reference genome*
ARI (Coverage 157X)	SPAdes (De novo)	69.5%
	EAGER (Reference-guided approach)	42.1%
	DACCOR (Hybrid approach with prior repeat Identification	96.8%

Table 4 Comparison between PT, PP index in time and space

Table 4 **Comparison between PT, PP index in time and space**

Metric	PT	PP
Space consumption	230 MB	298 MB
Time consumption	757 second	49 second