A Review on Machine-Learning and Nature-Inspired Algorithms for Genome Assembly

Asmae Yassine, Mohammed Essaid Riffi
LAROSERI Lab-Department of Computer Science, Chouaib Doukkali University, Morocco

Abstract—**Genome assembly plays a crucial role in the field of bioinformatics, as current sequencing technologies are unable to sequence an entire genome at once where the need for fragmenting into short sequences and reassembling them. The genomes often contain repetitive sequences and duplicated regions, which can lead to ambiguities during assembly. Thus, the process of reconstructing a complete genome from a set of reads necessitates the use of efficient assembly programs. Over time, as genome sequencing technology has advanced, the methods for genome assembly have also evolved, resulting in the utilization of various genome assemblers. Many artificial intelligence techniques such as machine learning and nature-inspired algorithms have been applied in genome assembly in recent years. These technologies have the potential to significantly enhance the accuracy of genome assembly, leading to functionally correct genome reconstructions. This review paper aims to provide an overview of the genome assembly, highlighting the significance of different methods used in machine learning techniques and nature-inspiring algorithms in achieving accurate and efficient genome assembly. By examining the advancements and possibilities brought about by different machine learning and metaheuristics approaches, this review paper offers insights into the future directions of genome assembly.**

Keywords*—*Artificial intelligence; genome assembly; machine learning; bioinformatics; bio-inspired algorithms

I. Introduction

DNA, short for Deoxyribonucleic acid, is a macromolecule that contains the genetic instructions for the development, functioning, and reproduction of all living organisms. It consists of two strands that form a double helix structure [1]. The four nucleotides—adenine (A), cytosine (C), guanine (G), and thymine (T)— make up the building blocks of DNA. The sequence of these nucleotides determines the genetic code. DNA carries the hereditary information passed from parents to offspring, containing genes that encode proteins and regulate various biological processes. DNA replication ensures that each new cell receives a complete copy of the genetic material during cell division.

Advances in DNA sequencing technologies have revolutionized biological research, enabling the analysis of DNA sequences on a large scale. DNA sequencing helps unravel the genetic code, identify mutations, study genetic variations, trace evolutionary relationships, and diagnose genetic diseases. DNA plays a central role in fields such as genomics, evolutionary biology, genetic engineering, forensic science, and medicine. It serves as a foundation for understanding the complexities of life, exploring the diversity of species, and developing innovative approaches [2] for disease treatment and prevention.

To determine the DNA material code within specific living organisms, the DNA sequencing is utilized for the identification and characterization of genes within an organism's genome. By determining the sequence of genes, researchers can analyze their functions, regulatory elements, and evolutionary history. Genome assembly is the computational process that comes after sequencing and aims to align reads of a DNA sequence into the correct order to reconstruct the original structure of the genome [3].

The novelty of this review stands out for its in-depth analysis of the most important advancements and challenges involved in genome assembly. By achieving a synthesis of findings from multiple research studies, the review presents a comprehensive survey of the current state of the machine-learning and nature-inspired genome assemblers, while also evaluating the recent bio-inspired techniques and providing recommendations for future research. This gives researchers insightful information and direction about the latest techniques and approaches used in this rapidly growing field. Thus, this study remains as an exceptional and valuable contribution to the existing literature.

The work in this paper is organized as follows: Section II presents an overview about the genome assembly. Section III describe machine-learning techniques used in DNA sequence assembly. Section IV introduce de novo assemblers. Section V review nature-inspired algorithms and metaheuristics solving the genome assembly problem and their best computational results. Section VI provide a discussion and analysis of the findings, proceeded by a conclusion in the last section.

II. Genome Assembly Overview

The DNA genome assembly is one of many bioinformatics problems that uses machine learning technologies and nature-inspired approaches and metaheuristics in aim to be solved. When sequencing the genomes of bacteria, viruses, or humans, this problem is extremely important; it occurs during the final stage of DNA reading, particularly for long strands of DNA. Large strands are repeatedly broken into several little fragments. After that a computer program will assemble together the fragments into a string that matches the original DNA sequence. Finding an organism's DNA sequence is helpful for both applied and fundamental study into how and why they live. Given the significance of DNA to living organisms, understanding a DNA sequence could be helpful in almost any biological study. For instance, it can be applied in medicine to locate, identify, and potentially develop cures for and genetic diseases. Similar to how pathogen research may result in medicines for infectious diseases and virus transmissions. Due to the rapid advancements in sequencing technologies and the increasing demand for sequencing services, various sequencing platforms have become extensively utilized in recent years. As a consequence, sequence assembly has emerged as a critical process with diverse applications in the field of bioinformatics [4]. Consequently, the significance of efficient DNA assemblers has escalated, as they play a pivotal role in reconstructing complete genomes from fragmented sequencing data

III. Machine Learning Techniques

McCarthy et al. suggested in the summer of 1956 in a conference at Dartmouth College, that computers could be programmed to think and reason. They referred to this idea as artificial intelligence (AI) [5] a field used to the simulation of human intelligence processes by computer systems. Machine learning (ML) is a specific method to achieve this goal which include deep learning (DL) and artificial neural networks (ANN) methods [6].

Machine learning is used in DNA sequence assembly for pre-grouping reads into similar groups before the assembly process is carried out and it was proven that is an alternate method for lowering the overall computational complexity of the genome assembly process. In [7] the authors suggested building a recurrent neural network (RNN), where the goal is to train the network to track the sequence of bases that constitute a given fragment and assign all of the sequences that are properly tracked by this network to the same cluster [7]. This machine learning method applies a three-layer RMLP (Recurrent Multi Layer Perceptron) neural network with five input nodes dedicated to the five possible incoming symbols (A, C, G, T, and N) in the first layer, a hidden layer of 27 nodes, and four output nodes for the output layer in the forecast of the next base of the sequence created by the network [7]. The experimental results obtained after comparing the proposed neural network method with another conventional algorithm have shown that if performed on multiprocessor machines, the proposed procedure may prove to be less expensive than ones that are currently used. Any assembly method may be employed in place of the CAP approach, which greatly improves its efficiency and yields superior outcomes. While employing various definitions of distance, the two approaches produce collections of fragments with more or less comparable properties. The authors also indicated [7] that their proposition is richer due to the fact that is based on the internal structure of the strings rather than just topological similarities.

The author [8] also proposed the construction of an artificial neural network based binning of reads to assist assembly process. After producing the required reads for assembly using a simulated shotgun sequencing technique. Four assembly techniques were then simulated : The greedy assembler, the de Bruijn assembler, the greedy neural network assembler, and the de Bruijn neural network assembler. In order to determine which of these assembly procedures provided the maximum coverage accuracy and with what level of computational complexity, simulations were performed. The research also looks into the advantages of combining the greedy and de Bruijn assembly algorithms with a "divide and conquer" strategy the greedy assembler's computational efficiency could be significantly improved. The author in [8] proposed in future work to investigate strategies of reducing the complexity associated with the training and grouping process and to take advantage of the parallelisable nature of the neural network grouping scheme.

The study in the paper [9] offers two techniques for separating sequencing method errors from natural variance. The first is an analytical technique for choosing appropriate error candidates. In order to identify the bases that are uncommon within fuzzy grouped clusters of closely related sequences, it applies similarity weights between pairs of sequences. The chosen candidates are then filtered using a classification model built using a suitable ML technique and rests on frequency vectors [9]. Both the usage of weights and the ML-based regrouping were proven to greatly reduce the set of potential errors, without missing simulated faults along the way for artificial neural networks and the Hoeffding tree classifier. The performance of the RIPPER rule learner and random forest classifier, however, drastically declined. On hexaploid wheat, the ML-based models showed raw accuracy high enough to imply that they might be utilized independently of the analytical approach for error discovery. According to the authors it was demonstrated that applying the ML model as an additional filter while calling variants significantly changes the results. Further validation were needed to validate the results achieved on hexaploid wheat.

IV. De-Novo Assemblers

Three major strategies are widely used for the novo assemblers in Bioinformatics to solve the DNA Assembly Problem: greedy graph-based algorithms, de Bruijn graphs, and the overlap-layout-consensus (OLC) approach mainly adopted by the nature-inspired algorithms presented in Section V.

De novo gene assemblers have no reference genome for assembling DNA sequences and they are classified into two types: Greedy-based De-novo assemblers Graph-method assemblers: String and DeBruijn. Several graph based common algorithms were used for genome sequence assembly CAP3 [10], PHRAP [11] , TIGR [12].

Authors in [13] suggested a novel DNA sequence assembly approach that combines the advantages of shotgun and sequencing by hybridization (SBH). The technique makes use of the high coverage and low error rates in sequencing made available by the development of efficient DNA sequencing machines. The authors were proceeding in the development of full software with all the characteristics stated in [13] .They created a prototype that incorporates some of the algorithm's fundamental components, and utilized this prototype to put together synthetic sequencing data. They presented results of such an experiment, mostly as a demonstration of concept for their methodology and based on their preliminary investigations [13], the algorithm promises to be very fast and practical for DNA sequence assembly.

A novel EULER algorithm have been designed in [14] and for the first time fixes the repeat problem in fragment assembly. The primaly goal of the authors was the fragment assembly's reduction to a variant of the standard Eulerian path problem, which makes it possible to produce precise answers to complex sequencing issues. In contrast to the CELERA assembler [15], EULER utilizes such repeats as a strong fragment assembly tool rather than masking them. Based on the de Bruijn graph concept. In order to explain their method, they adopted the DNA sequence as a thread with repeated sections attached together by glue. Every repeat in the resulting de Brujin graph having five edges [14], corresponds to one edge rather than a group of vertices in the layout graph.

In [16] the authors presented the ARACHNE computer system, which uses paired-end whole-genome shotgun reads to assemble genome sequences. The key characteristics of ARACHNE include an effective and sensitive method for detecting read overlaps, a method for scoring overlaps that achieves high accuracy by fixing errors prior to assembly, read merger using forward-reverse links, and identifying repeat contigs by forward-reverse link inconsistency.

ARACHNE begins by identifying and aligning overlaps, or pairs of reads that appear to be overlapping. In later rounds, some of these false overlaps caused by repeated sequences in the genome will be removed [16]. An effective overlap detection is achieved. The program utilizes a sort and extend technique that scales about linearly rather than comparing every pair of reads. This method entails creating a sorted table of each k-letter (k-mer) and its source so that related k-mers appear successively. The algorithm next eliminates highly frequent k-mers, which often correlate to highcopy, high-fidelity repetitive sequences in the genome, in order to improve the effectiveness of the overlap detection procedure. After identifying all read pairs that share one or more overlapping k-mer, the algorithm applies a three-step procedure to effectively align the reads. First, overlapping shared k-mers are combined. Next, shared k-mers are extended to alignments. Finally, dynamic programming is used to enhance the alignments. In a similar manner, ARACHNE corrects random insertions and deletions caused by apparent sequencing errors [16]. The alignments are modified in accordance with how the reads are adjusted.

Simulated reads that covered many genomes order have been generated to evaluate ARACHNE. These simulated reads' assemblies produced virtually full coverage of the corresponding genomes, with a few contigs combined into even fewer supercontigs (or scaffolds) [16]. Contig coverage after genome assembly ranged between 97 and 98 percent, with at least 92 percent of the reads being used in every case. For full coverage, the N50 contig length is 350 kb, whereas for half coverage, it is 17 kb. The length of the N50 supercontigs varies significantly within the genomes.

The authors precised that assembly accuracy was good, but it wasn't perfect due the fact that there were a very small number of additional misassemblies and little errors happened about once every 1 Mb. Assembling the Drosophila genome was quick, requiring only 21 hours on 8.4 Gb of RAM in a single 667 MHz processor.

In this work [17], the authors developed Velvet a novel collection of de Bruijn graph-based sequence assembly methods for very small reads. The main objective of the approach is both remove errors and resolve large number of repeats repeats in the presence of pair read information. The error correction technique initially merges sequences that belong together, and the repetition solver then separates path that share local overlaps. Authors have evaluated Velvet using both simulated and real data [17]. The algorithm has the potential of assembling bacterial genomes with N50 contig lengths of up to 50 kb and simulations on 5-Mb portions of large mammalian genomes with contigs of 3 kb using only relatively small paired simulated reads. The two other short read assemblers, SSAKE [18] and VCAKE [19], were compared to Velvet. The algorithms are different from one another mostly in how they handle errors. By looking for reads in a hash table, SSAKE and VCAKE automatically explore a de Bruijn graph in a step-by-step manner. Velvet is considerably faster and generates larger contigs without misassembly, but it takes a little bit more memory. Furthermore, it has great precision and covers a significant part of the genome. The authors attempted to use EULER [14] and SHARCGS [20], but the tools were unable to handle their data sets. According to authors, this was most likely because the differed expected input, notably in terms of covering depth and read lenght.

Building on earlier works [21] [14] the authors in [22] developed MULTIBRIDGING a de Brujin graph based assembly algorithm for shotgun sequencing under the criterion of complete reconstruction, which can achieve very close to the lower bound for repeat statistics of a variety of sequenced genomes, including the GAGE datasets. As results assembling the repeat statistics of hc19, have shown successful reassembling desired with probability 99%.

The La Jolla Assembler (LJA) [23] a fast algorithm with three modules that address the three challenges in assembling HiFi reads: jumboDBG (constructing large de Bruijn graphs), multiplexDBG (using the entire read-length for resolving repeats), and mowerDBG (error-correcting reads), was designed to enable automated assemblies of long, HiFi reads. The Bloom filter [24], sparse de Bruijn graphs [25], disjoint generation [26] and rolling hash [27] were all used in the jumboDBG approach. LJA builds the de Bruijn graph for huge genomes and large k-mer sizes and turns it into a multiplex de Bruijn graph with changing k-mer sizes, reducing the error rate in HiFi reads by three orders of magnitude. The suggested approach not only produces five times fewer misassemblies than state-of-the-art assemblers, but also generates more contiguous assemblies. In the publication, the automated assembly of a human genome which successfully assembled all six chromosomes was used to illustrate the usefulness of LJA.

V. Nature-inspired Algorithms for Genome Assembly

Nature-inspired optimization algorithms is defined as a group of algorithms that are inspired by the behavior natural systems, including bio-inspired algorithms , swarm intelligence and evolutionary algorithms. Inspired by animal, insects behaviors, biology and chemical reactions, those algorithms have provided many engineering, medical and bioinformatics solutions such as solving the DNA Fragment Assembly Problem. The genetic assembly is a critical step in any genomic project, it attempts in reconstructing a DNA sequence from a set of a large number of fragments taken obtained by biologists in the laboratory. DNA Fragment Assembly Problem is known to be an NP-hard combinatorial optimization problem, therefore efficient approximate metaheuristics are required to solve such kinds of problems. The purpose of the study presented in this section is to analyze and synthesize the existing nature-inspired optimization algorithms in for genome assembly. Since the genome assembly is a particularly difficult problem in computational biology due to the problem's NP-hardness, the ideal solution cannot be found. So, it necessitates the use of metaheuristics and other computational techniques of intermediate complexity. Which their goal is to compare all possible solutions to an optimization problem in order to choose the best (feasible) one. They evaluate prospective solutions and perform a number of operations on them in an effort to find better alternatives in order to accomplish this.

As genome assembly is a combinatorial optimization problem, different nature inspired algorithms and metaheuristics have been proposed in past few decades to solve this problem. The process of these metaheuristics involves the reconstruction of the original genome sequence from a set of fragments (reads) aligned in a correct order by exploring a large solution space. Numbers from 1 to N are assigned to the set of fragments, where N represents the total number of fragments. By reordering this list of numbers using the fitness function, the algorithm aims to find the optimal order that reconstructs the complete genome sequence through a process of iterative optimization.

Generally an incremental solution of a metaheuristic algorithm for DNA fragment assembly is described as follows: The algorithm starts by setting up parameters and creating an initial assembly of DNA fragments. To assess the quality of the assembly, a fitness function is established. In the case of genome assembly the fitness function involves maximizing the fragment's scores obtained through semi-global alignment of the DNA fragments. The metaheuristic algorithm proceeds with iterations, each aimed at optimizing the assembly step by step. In each iteration, the current assembly is perturbed to explore neighboring solutions in the search space. The algorithm decides whether to accept the newly generated solution or keep the previous one based on a specific acceptance criterion for each iteration. As the algorithm progresses through iterations, the incremental assembly continues to integrate improvements made in previous steps. The process is considered final when the algorithm reaches a near-optimal solution or fulfills the desired quality standards, the stopping criterion is based on predefined conditions, such as a fixed number of iterations or convergence of the fitness function.

Many Swarm Optimization algorithms have evolved the fragment assembly problem: [28], [29], [30], [31] Cuckoo Search algorithm [33] Harmony Search algorithm [34] hybrid crow search algorithm [35], Cat Swarm Optimization [36], etc. A total of 30 publications on nature-inspired optimization algorithms dealing with DNA genome assembly have been reviewed as shown in Table I.

A. Swarm Intelligence Algorithms

Verma et al. have proposed the DSAPSO method [28] to resolve the DNA sequence assembly problem using Particle Swarm Optimization (PSO) with Shortest Position Value (SPV) rule. To convert the continuous version of PSO to discrete version SPV rule is used in solving the DNA sequence assembly problem wich is a discret problem. The proposed methodology outperforms the genetic algorithm (GA) for every DNA data set, according to the results of a comparison of the DSAPSO results with those of the GA.

By maximizing the overlapping-score measurement, a hybrid particle swarm optimization VNS-based local search approach for solving the DNA fragment assembly (DFA) problem is proposed in [31]. To make PSO appropriate for DFA, the particles are encoded using the lowest position value (SPV) rule. During the PSO search process, VNS local search is used to enhance the quality of the globally best solution generated from the PSO algorithm. The results demonstrated that the algorithm can significantly outperform other PSO-based algorithms with different-sized benchmarks in terms of overlap score.

In other research [32], the same authors suggests a novel memetic GSA method called MGSA in order to solve DNA FAP problem. The overlap-layout-consensus model known as MGSA is based on tabu search for population initialization. This algorithm uses an SPV rule to convert continuous position values into job sequences, initializes the population with a tabu search, and then uses simulated annealing with VNS as the local search method to improve the quality of the best global solution produced by the GSA algorithm.These modifications create a balance between exploitation and exploration. The simulation results show that the algorithm MGSA maximizes the overlap score of 19 benchmark instances, however its disadvantage is requiring more processing time than the current techniques. In order to resolve this issue, the authors want to take into account DNA sequence compression, fuzzy entropy and adapting the MGSA strategy to the de-Bruijn-graph (DBG) model in future works in order to decrease the computation time.

Adaptive Particle Swarm Optimization was proposed in [30] and the experimental results of study on the impact of inertia weight and the cognitive and social components for enhancing the PSO efficiency to obtain the optimal fitness score, were presented with the simulation of three methods: The PSO with constant inertia weight (CIW), PSO with dynamically varying inertia weight (DVIW) and APSO.

The paper [37] presents a new particle swarm optimization and differential evolution approach using the SPV rule to convert the continuous variables used in PSO and DE to the permutation required to solve the FAP. The authors have conducted four different experiments The purpose of Experiment 1 was to evaluate the PPSO+DE algorithm with those that performed the best on the sixteen typical benchmarks. The Lin-Kernighan method was used in Experiment 2 to tackle each of the sixteen benchmark issues utilizing the TSP approach. The results of the Lin-Kernighan algorithm were used to determine the best solutions in Experiment 3. In Experiment 4, the Staphylococcus aureus COL Main Chromosome test data was used to test the TSP technique.

In this study [38], the authors suggested a new approach to solve the sequence assembly problem using Particle Swarm optimization (PSO) with Naive crossover and shortest position value (SPV) rule. There are two phases in PSO with Naive Crossover with SPV: The initialization phase, where individuals are initialized, and the update phase, where new solutions are generated and updated. The real coded values are converted to discrete values using SPV rules. According to the authors, the DNA sequence assembly utilizing PSO algorithm with naïve crossover (DSAPSONC) has demonstrated the effectiveness in solving the sequence assembly problem.

In order to effectively solve the fragment assembly problem, a new DPSO method that operates directly in the search space of permutations has been proposed [39]. The Probabilistic Edge Recombination (PER) operator is the main element of the suggested approach. Through the probabilistic recombination of edges connecting adges from current position, the personal best, and the group best, this operator creates an alternate position. In this probabilistic construction, the utilization of overlap lengths between fragments has also been taken into consideration. With this purpose, memetic algorithms with a new fast variant local search of PALS known as quick-PALS were developed to improve the intensification potential. To show the efficiency and potency of the employed algorithms, two sets of validation experiments have been performed. The authors claim that when compared to present in litterature assembly techniques, the algorithms performed better.

A new PSO variation was suggested in [40] that uses chaos, levy flight, and adaptive parameters to solve the genome sequencing problem which is transformed into a discrete optimization problem while using the SPV rule. The proposed algorithm incorporates chaos in two distinct ways: applying chaotic inertia weight and chaotic initialization. To ensure the balance between exploitation, which is encouraged by a lower inertia weight, and exploration, which is encouraged by a bigger inertia weight, a chaotic inertia weight was applied. The production of particles through using Levy, chaos and refinement Flight ensure that the particles are initialized with a high fitness score. The chaotic initialization provides favorable circumstances for discovering better values [40]. For the four datasets studied in the paper, the Chaotic Particle Swarm Optimization with Levy Flight performs better than other PSO variations by 7% to 24%. When compared to other algorithms on the basis of Standard Deviation, the proposed approach does not, however, demonstrate a significant improvement. It has a higher ranking but inconsistent performance. To solve this issue the authors suggests to used alternative PSO algorithm versions in future.

This study [33] presents the Cuckoo Search Algorithm (CS) as a novel optimization approach for genome sequence assembly, inspired by the behavior of cuckoos. CS incorporates levy flight and brood parasitic behavior, mimicking the process of cuckoos laying eggs in host nests. The algorithm is a population-based search procedure widely applicable to complex optimization problems. These birds lay their eggs in the nests of other birds and use various strategies to increase the chances of their eggs hatching. The algorithm models this behavior by representing each solution as an egg and using Levy flight to generate new solutions. The algorithm follows three main rules: -Each cuckoo lays one egg at a time in a random nest -Only the nests with high-quality eggs are preserved, - Hosts have a probability of discovering alien eggs and can either remove them or abandon their nests. Various algorithm settings are analyzed to determine the most effective configuration, and CS's efficiency is evaluated against PSO and its variants.

In this study [41], a hybrid cuckoo-search genetic algorithm (CSGA) was suggested as a nature-inspired swarm optimization algorithm. The total assembly time and the number of reorientations during the assembly process are taken into account by the cost criterion for optimization. An example assembly with 19 components has been shown to demonstrate how the CSGA is applied, and the results have been compared with those of the Genetic Algorithm (GA). According to the findings, the CSGA algorithm not only generates optimal assembly sequences for the given problem at costs equivalent to those of GA, but it has also been discovered to have a faster convergence rate than GA.

The study [42] introduces subsequence-based matching techniques using the CS and PSO algorithms. These methods were implemented in Java and utilized MapReduce for Hadoop. The experimental results validate the effectiveness of the proposed techniques, showcasing their ability to achieve extensive DNA fragment coverage and high matching accuracy. Furthermore, the performance analysis reveals that the CS algorithm outperforms the PSO algorithm in terms of overall performance.

B. Simulated Annealing based Algorithms

Simulated annealing is a computing technique that seeks for the optimal solution by using randomness. It's inspired from a related technique called Annealing in Metallurgy which mimics the physical solid annealing process where a glass material or metal is heated to a high temperature and then allowed to cool. The authors in [43], [44] have introduced and applied methods solving the DNA fragment assembly problem with the addition of the inversion and transposition operators to a simulated annealer by [45] the performance have successfully been increased. These studies generally raise a variety of other important issues, particularly those relating to the significance of solution space redundancy and the synergistic interactions between the various operators [46].

In this paper [47], a parallel models of Simulated Annealing (SA) was proposed combined with Genetic Algorithm (GA) for solving the DNA fragment assembly problem. They employed SA as a local search method within the GA framework. The experimental results demonstrate that the parallel approach improves the quality of solutions while reducing the overall runtime. Comparing the execution times, SA outperforms GA by being faster. However, SA tends to produce worse fitness values compared to GA In this paper [48], Simulated Annealing-based local search have been utilized to improve the final solution obtained by the Chemical Reaction Optimisation (CRO) algorithm in solving the DNA fragment assembly problem. The CRO approach is used in seeking for the best layout where the objective function is minimized. The main process of the algorithm starts with setting valued to the control parameters after the determination of the initial population. Then one of the four collisions of the CRO algorithm is performed in each iteration. After any new minimum fitness value is checked and saved. After that, to retain the diversity of the population the worst 20 percent of the population is replaced with new solutions. When no amelioration has occured, the simulated annealing method is used in order to enhance the best solution found.

SA maintains at the same temperature for a period of time while a predetermined number of iterations are set. Then, the heat becomes colder. One of the following three operators is chosen for each iteration [48]:

1) Inversion: Two points are chosen at random for this operator [48]. Then, between them, the order of the fragments is reversed.

2) Specific inversion: One contig's orientation is reversed. To do this, a permutation point is randomly chosen, and the contig containing this fragment is identified. The fragments in the chosen contig are then rearranged in reverse order [48].

3) Transposition: This operator [48] shifts a contig to a new position between two points in different distinct nearby contigs that are selected for the contig movement.

The experimental results in the paper [48] have shown that combining CRO and SA have lead to the highest overlap scores.

C. Local Search based Algorithms

In this paper [49], the authors have presented PALS (Problem Aware Local Search) a fast and accurate local search algorithm that, after comparing its results with commercially available assemblers Phrap and CAP3, pattern matching algorithms (PMA) and genetic algorithms (GA), it have shown to be competitive against those present specialized assemblers. They also explored the effect of many alternative approaches on the efficacy of the suggested algorithm. The main objective of PALS algorithm is to obtain one single contig by finding a fragment's order that minimizes the number of contigs, which is different from the other assembling algorithms that aim to search for solutions having in the layout maximum overlap between adjacent fragments . The three main methods of the PALS algorithm are [49]:

- GenerateInitialSolution method: Generates a single solution (successive overlapping fragments) and is continually updated by the application of ordered movements.
- ApplyMovement method: Makes a movement perturbation and alters the subpermutation between to positions i and j.
- CalculateDelta method: Calculates the variation in the overlap and in the number of contigs this method is considered the main step of the PALS algorithm.

In another paper [50] two changes were proposed to the principal PALS: The first goal is to avoid the local optima and premature convergence. In this case, the method for choosing the enhancing perturbation to be applied to the existing solution at each algorithmic step is changed in a way that leads to a significant improvement. The second one is to minimize the computational demands of the algorithm, this change involves applying multiple independent perturbations rather than a single perturbation to enhance the present solution at each algorithmic step.

The authors in [50] have noticed that the optimization of the fitness is not the same as the optimization of the number of contigs which is the present objective although the two goals are complementary. As a result, the search mechanism in PALS includes an estimation of the number of contigs. It selects the movement with the lowest variation of contigs to orient the search towards solutions that improve the number of contigs which is the movement that reduces or maintains the number of contigs.

Authors in [50] have suggested changing the movement selection technique to prevent the premature convergence. With the modified PALS known as PALS2, the movement with the lowest contigs variation have always been selected, but in contrast to the main PALS, in the situation that there are several movements with the same contig variation, the movement with the lowest fitness variation is chosen.

To avoid the significant amount of recalculations required in each step in the process of PALS where only one single movement in every step is applied, the authors have proposed the algorithm PALS2-many [50] where many movements are used in each step by developing a second variant in PALS2 in order to improve the solution

The paper [51] have presented a discrete whale optimization algorithm (DWOA) modeling the approach taken by humpback whales when looking for victim or prey by employing conventional operators adapted from evolutionary algorithms. The whales assault their victim or prey using a remarkable feeding technique known as the bubble-net approach. They swim up to the surface after constricting loop after spiraling around the victim [52]. In order to avoid reducing the variation in the population, the whale positions used to look for the prey were produced randomly from the fragment numbers rather than utilizing a random whale. To show how effective DWOA is in converting continuous whale behaviors to discrete ones, it was compared to various WOA, DE, and SCA methods. The paper have also demonstrated the performance of the DWOA over those algorithms. To enhance the performance of the proposed Discrete Whale Optimization Algorithm (DWOA) in terms of fragment order, a local search technique called PALS2-many was incorporated [51]. This approach, known as DWOA-LS, combines the benefits of both DWOA and local search to optimize the fragment order. By integrating the local search, DWOA-LS not only maximizes the overlap score among the fragments but also minimizes the number of contigs, resulting in improved overall performance.

D. Genetic Algorithms

Genetic algorithm have also been applied in the DNA fragment assembly problem by [46]. The authors investigated various evolutionary algorithm operators for the issue and discovered that using "macro-operators" considerably boosts performance by exploiting fragment construction at gradually higher levels.

Individuals are the population of potential solutions that genetic algorithms operate on [53] [54] [55]. Usually, random people are used to initialize the population. Depending on their relative fitness, individuals are then either removed from the population or reproduced within it. Different operators are applied to the existing population of individuals to create new individuals ones. A generation is a group of people in any consecutive population. Typically, the genetic algorithm [46] [56] processes in the following order:

1) The algorithm creates a pool of solutions at random where random individuals are used to initialize the population.

2) It uses a fitness function for superior solutions selection. The fitness of each individual is evaluated during the selection process. Based on fitness, individuals are reproduced (copied) in different ways. Various genetic algorithms implement the concept of differential reproduction using various techniques [46]. However Parsons et al. have used generational genetic algorithm where a A new population is formed at each generation, totally replacing the prior population. Low fitness individuals have a low likelihood of being copied into the next generation, whereas high fitness individuals have a high likelihood of having several copies in the following generation.

3) To produce next-generation solutions, crossover and mutation processes are applied to successful solutions [56]. The authors have discussed in [46] all the crossover operators where the crossover rate indicates the average percentage of new people created by crossover per generation and defined the crossover as the selection of two individuals from the population, and the swapping of substrings from corresponding regions within the individuals. At the next generation, one or both of the new individuals are incorporated into the population. The operator's goal is to let incomplete solutions develop on various individuals before combining them to create a better solution.

By modifying a basic component of an individual, a mutation modifies that individual is the definitions of mutation in [46]. The chances that any element in an individual will change depends on the mutation rate. The resultant individual takes the place of the mutation's parent. It is thought by authors that mutation is successful because it both explores the search space close to existing individuals and saves solution components that have been totally excluded from the population by selection for the following generations [46]. Noting that in genetic algorithm the individuals are the fragment orders representing the DNA sequence solutions.

A new Hybrid genetic algorithm (GA+PALS) have been presented in the paper [57], the genetic algorithm have been used with PALS that was utilized as a mutation operator.The authors have compared the hybrid method with the original GA and PALS methods.A very effective assembler that enables the search of optimal solutions for numerous instances of this problem was obtained as a result. Authors have used the conventional recombination and mutation operators,then some solutions with a low probability, are randomly chosen from the existing offspring and enhanced utilizing the local search algorithm in the method's main loop. This type of hybridization is justified by the fact that, while the GA identifies good positions of the search space , PALS facilitates exploitation in the best regions discovered by its collaborator. Obviously, the goal in this situation is to determine whether we can develop another heuristic from the best of the two (the GA and PALS algorithm) that would outperform either of the two methods from which it was derived.

The authors have introduced two novel algorithms in recent publications: the Recentering-Restarting Genetic Algorithm (RRGA) and the Recentering-Restarting Hybrid Genetic Algorithm (RRHGA), as mentioned in the papers [58] and [59]respectively. The primary advantage emphasized in the paper [58] for RRGA is its ability to avoid local optima by exploring the search space and leveraging specific dynamic representations. Before initiating the algorithm, a center or reference point, representing a potential solution, is selected either through seeding or random selection. Uzma and Halim [59] suggest that starting with a solid solution is preferable, while RRGA refines the potential solutions. Once the center is determined, the population is generated.

In the direct representation approach, each individual in the population is created by applying a sequence of n transpositions to the center of the population. The ordered lists of fragments are modified through evolution, as explained in the work by [59]. The RRGA algorithm incorporates the Power Aware Local Search (PALS) operator as an evolutionary operator. The performance of the proposed algorithm is evaluated based on overlap scores and the quantity of contigs.

According to the authors, the initial arrangement of fragment orders in RRGA is known as the center, which represents the default arrangement of the dataset's fragments. The center is then optimized using a 2-opt heuristic. This process generates a set of chromosomes, from which the best chromosome is selected based on its fitness value. A comparison is made between the fitness value of the best chromosome and the center. If the fitness value of the best chromosome surpasses that of the center, the number of transpositions is reduced by five percent, and the center is replaced with the best chromosome.

To evaluate their work, the authors conducted three types of experiments. In the first set of experiments, the PALS operator was used as a genetic operator. In the second set, PALS was applied after running the Genetic Algorithm (GA). Finally, in the third set, PALS served as a genetic operator and was also used after the execution of the GA. The experiments were performed with and without force recentre methodologies, and the results were compared with the Recentering-Restarting Genetic Algorithm, PALS, Genetic Algorithm, and Hybrid Genetic Algorithm. The RRHGA approach demonstrated superior performance across all of these methods.

The paper [60] discusses the importance of studying genetic algorithms to address the DNA fragment assembly problem. The efficient GA operators that have proven successful in the TSP and QAP contexts served as the inspiration for the construction of a GA platform to tackle the DNA FAP problem. By identifying commonalities between the three DNA FAP, TSP, and QAP problems these efficient GA operators successfully been identified and integrated in the platform.

By carefully combining various GAs operators of the platform, an effective GA variant was created in this research [61] . In order to do that, various GAs operators have been studied in solving the DNA FAP problem [60]. The performance of these operators in the contexts of the TSP and QAP issues is already established, and this study has the advantage of comparing the GA results with the results of other previous GA studies in the context of the overlap score. The SCX crossover, a smart crossover that has never been utilized with DNA FAP, provided better results than the other crossover types under consideration, which is the study's most evident and important discovery. The best-designed GA variant outperformed the current GA algorithms at solving the DNA FAP problem and showed a notable improvement in accuracy with good and competitive results. This paper [61] is the first study to solve the DNA FAP problem form this perspective.

TABLE I. NATURE-INSPIRED ALGORITHMS IN LITERATURE

Inspiration	Stand alone algorithm	Hybridization algorithm	Total
Insects	[62] [66]	[65] [67] [63] [64]	6
Birds	[30] [33] [42] [28] [38] [39] [40]	[31] [32] [37] [41] [35]	12
Evolutionary	[46] [58] [61]	[57] [59]	5
Mammals	[36]	[51]	2
Annealing in metallurgy	[43][45][44] [47]	[48]	5
Total	17	13	30

E. Ant Colony Algorithms

This paper [62] introduces the application of an ant colony system algorithm for DNA fragment assembly. The proposed approach utilizes an asymmetric ordering representation, where the collective path generated by the ant colony represents the search solution. The study investigates two types of assembly problems: single-contig and multiple-contig problems. The simulation results demonstrate that, for single-contig problems, the ant colony system algorithm performs comparably to a nearest neighbor heuristic algorithm. However, in the case of multiple-contig problems, the ant colony system algorithm surpasses the nearest neighbor heuristic algorithm.

In [63] and [64] the ant colony system (ACS) algorithm was combined with the nearest neighbor heuristic (NNH) algorithm for solving the DNA fragment assembly. The ACS algorithm is utilized to create an optimized ordering sequence for the fragments, while the resulting contigs are assembled using the NNH rule. To evaluate its effectiveness, the ACS+NNH procedure is compared to the standard sequence assembly program CAP3. The results indicate that the overall performance of the combined ACS/NNH technique surpasses that of CAP3. Specifically, when dealing with large problem sizes, the ACS/NNH solutions exhibit higher quality than the CAP3 solutions. It is observed that CAP3 tends to generate a greater number of contigs compared to the ACS+NNH procedure. Thus, the combined ACS/NNH approach demonstrates superior performance and improved contig quality, particularly for larger-scale problems, as opposed to the CAP3 program.

F. Bee Algorithms

The nature-inspired Bee Colony metaheuristic algorithms are population-based search algorithms based on various biological and natural processes observed in the food foraging behaviour of honey bee colonies. In [65] the authors have designed two different Bee algorithms: Artificial Bee Colony (ABC) Algorithm and Queen-bee Evaluation Based On Genetic Algorithm (QEGA).

Artificial bee colony (ABC) algorithm was developed using the notion of the honey bee swarm's intelligent behavior. To create new effective search algorithms, honey bees use strategies like the waggle dance [65]. Three types of bees that compose the artificial bee colony in the ABC algorithm are workers or employed, onlookers, and scouts. A bee that waits at the dance area to decide which food source to choose, representing one possible solution of a permutation of DNA sequence fragments based on the waggle dance of the employed bee, is referred to as an onlooker. And a bee that moves to the food source that had previously visited is referred to as a worker bee. A scout bee is one that hunts for food at random.The nectar content of a food source represents the DNA assembly problem's fitness solution.

In the ABC algorithm [65], the employed artificial bees make up the first half of the colony, while the observers make up the second half. When both the employed and onlookers bees have consumed the employed bee's food supply, it turns into a scout. A worker or onlooker bee modifies the solution with PALS method for the locating a new food source and evaluates its nectar by calculating the fitness value of the new solution.

In the other category of Bee Colony algorithms, authors [65] designed Queen-bee evolution based on genetic algorithm (QEGA), which was inspired by the queen bee evolution process and has been utilized to improve the optimization capabilities of genetic algorithms in solving the DNA FAP. Genetic algorithms are capable of reaching the global optimum quickly due to the queen-bee evolution, which also reduces the risk of premature convergence. The authors have utilized problem aware local search (PALS) for an effective mutation.

G. Firefly Algorithm

The Firefly Algorithm (FA), a population-based algorithm inspired by the behavior and lighting patterns of fireflies created by Yang [66], is a recent nature-inspired algorithm that has excelled in many number of fields.The firefly have the following attributes according to Yang's theory:

1) As all fireflies are not gender-specific, they are attracted to each other regardless of their gender orientation.

2) Their brightness is inversely correlated with attractiveness. The less brilliant firefly will therefore travel toward the brighter one for any pair of flashing fireflies. When their distance grows, their attractiveness decrease. Hence, when there is no distance between two fireflies, the attractiveness is equal to the brightness. If none can see a better firefly it will move at random.

3) The environment of the objective function influences or determines a firefly's radiance.

Authors in the paper [67] have designed Discrete Firefly Algorithm design for Graphics Processing Units (GPU-DFA) and analyzed it behaviour to solve the DNA assembly problem. The main objective of the authors of the paper while designing the algorithm GPU-DFA is to establish an efficient model that runs the main processes of DFA entirely on GPU so the algorithm can support large numbers of fireflies due to optimized data-structures. The initialization and evaluation of each solution are completed one at a time in the paper's GPU-DFA method. DNA pieces are randomly permuted to produce each firefly i. The firefly I is then assessed, and its brightness (fitness) is determined. In order to compute them, GPU-DFA uses parallel threads. Several consecutive threads can make use of different memory space.

TABLE II. Comparison of the Fitness Value of the Studied Algorithms from [48], [67], [35], [51],[59], [37], [65], [39] on Different Instances

Dataset	CRO+SA [48]	GPU-DFA+LS [67]	CSA-P2M*Fit [35]	DWOA-LS [51]	RRHGA [59]	PPSO+DE [37]	QEGA [65]	PER-PSO-(hi)-ls [39]
$M15421_5$	38746	38746	38746	38746	22598	38686	38578	38746
$M15421_6$	48052	48048	48052	48052	29469	47669	47882	48052
$M15421_7$	55171	55072	55171	55171	32744	54891	55020	55171
$J02459_7$	116700	116700	116700	116700	68736	114381	116222	116700
$BX842596_4$	227914	227233	227920	227920	125711	224797	227252	227920
$BX842596_7$	444518	444162	445422	445422	247856	429338	443600	445422

H. Crow Search Algorithms

A novel crow search inspired algorithm (CSA) was proposed [35] to solve the DNA fragment assembly problem following the OLC model. Crows represent individuals in the population. Each crow maintains a unique hiding place, analogous to a solution candidate in the DNA fragment assembly problem. To protect their hiding places, crows employ specific defensive measures against potential followers, This behavior is presented through the following descriptions: -Crows live in social groups known as flocks. -Each crow maintains a memory of the location of its own hiding place. -Crows engage in a follow-the-leader strategy to identify the hiding places of other crows. -A crow defends its hiding place from potential attackers by employing a probabilistic defense mechanism. Since the FAP is a discrete problem, and the original algorithm was designed for continuous optimization problems, Allaoui et al. proposed using a modified version of the ordered crossover operator (OX). CSA was also combined with a local search method and utilized standard operators from evolutionary algorithms. The resulting approach, CSA-P2M *Fit Algorithm, outperformed other algorithms designed for the same purpose. It demonstrated accelerated search and yielded high-quality solutions in the context of DNA fragment assembly.

I. Cat Swarm Optimization

Recently, Yassine et al. presented in [36] the application of the Cat Swarm Optimization algorithm (CSO) in the DNA fragment assembly problem. This metaheuristic is a swarm intelligence algorithm that incorporates the natural behavior of cats and takes inspiration from the characteristics of cats, which are typically lazy creatures that spend a significant amount of time resting in a seeking mode. However, even during their resting periods, they remain aware of their surroundings. When cats sense a target, they switch to a tracing mode and start moving towards it.

In CSO, each cat within the swarm is represented by its position, velocity, and a flag indicating whether it is in seeking mode or tracing mode. The position of a cat corresponds to a potential solution to the problem being optimized. The velocity of the cat influences its movement within the search space. The flag determines the current mode of the cat, indicating whether it is in seeking mode (resting) or tracing mode (actively moving towards a target). The mixing ratio (MR) is a parameter in CSO that determines in which mode the cat will go into.

By simulating the natural behavior of cats and incorporating it into an optimization algorithm, CSO aims to find accurate solutions to the DNA fragment assembly problem by efficiently exploring the search space. The balancing of seeking and tracing modes through the mixing ratio enables the algorithm to adapt its exploration and exploitation strategies based on the problem characteristics and the current state of the swarm.

The Table II synthesize eight main algorithms hybridized with local search and other methods for solving the DNA fragment assembly problem. The first column of the table presents the dataset instances names provided from [68]: M15421(5), M15421(6) and M15421(7) from the human apolipoprotein B gene. j02459(7) instance from Complete nucleotide sequence of the cohesive ends of bacteriophage lambda DNA. The two instances bx842596(4) and bx842596(7) from Neurospora crassa DNA linkage group II BAC clone B10K17. In the other columns the fitness results obtained by the algorithms are presented: CRO+SA (Chemical Reaction Optimisation) [48], GPU-DFA+LS (Discrete Firefly Algorithm design for Graphics Processing Units) [67], CSA-P2M*Fit (Crow Search Algorithm and ALS2-many) [35], DWOA-LS (Discrete Whale Optimization Algorithm PALS2-many) [51], RRHGA (Recentering–Restarting Hybrid Genetic Algorithm) [59], PPSO+DE (Parallel Particle Swarm Optimization and Differential Evolution) [37], QEGA (Queen-bee Evaluation Based On Genetic Algorithm)[65], PER-PSO-(hi)-ls (Probabilistic Edge Recombination Particle Swarm Optimization and quick-PALS) [39].

It's clear seen from the Table II that the hybrid methods CSA-P2M*Fit, DWOA-LS and PER-PSO-(hi)-ls outperfomed in all the instances. CRO+SA give better results in M15421(5), M15421(6), M15421(7) and j02459(7). GPU-DFA+LS showed high fitness values too in M15421(5) and j02459(7).

However, it is important to consider that every metaheuristic algorithm possesses certain parameters that contribute to enhancing the algorithm's results. In most of the algorithms, the different parameters settings were applied in different test experiments and were varying for each instance of the dataset.

VI. Discussion

The field of genome assembly algorithms has experienced rapid and exponential growth. Over time, there has been a significant increase in the development and advancement of these algorithms. This growth can be attributed to several factors, including the availability of high-throughput sequencing technologies, the decreasing cost of sequencing, and the increasing demand for accurate and complete genome assemblies. Genome assembly algorithms play a crucial role in reconstructing the fragmented DNA sequences obtained from sequencing machines into complete genomes. As the complexity and size of genomes vary across different organisms, the development of efficient and accurate assembly algorithms has become essential. Advancements in assembly algorithms have been driven by a combination of algorithmic innovations, computational resources, and improved understanding of the characteristics of DNA sequences. Researchers have developed various algorithmic approaches, including machine learning methods, nature inspired metaheuristics based on overlap-layout-consensus and de Bruijn graph-based approaches, and hybrid methods that combine multiple strategies.

As a result of these combined factors, the field of genome assembly algorithms has experienced remarkable growth, with continuous improvements in scalability, computational efficiency and in assembly quality [69]. This ongoing progress in algorithm development is crucial for advancing genomics research, enabling discoveries, and understanding the complexities of genomes across different species. This review paper provide collaboration and knowledge exchange among researchers, enabling them to overcome the genome assembly challenges through the application of machine learning and nature-inspired optimization algorithms. Machine learning methods and metaheuristic methods are known to be two distinct approaches used in problem-solving domains including genome assembly. Each approach has its strengths and limitations, and a comparison between the two can provide insights into their applicability and effectiveness in different scenarios. Methods provided by machine learning, such as supervised learning, unsupervised learning, and reinforcement learning, utilize algorithms that learn patterns and relationships from data. These methods excel in tasks where large amounts of labeled or unlabeled data are available. In genome assembly, machine learning methods can be employed for various purposes, such as error correction, read alignment, and sequence classification. They can leverage the inherent structure and patterns within the genomic data to make predictions and improve assembly accuracy. However, machine learning methods often require extensive training data and may be computationally intensive, especially for complex problems with high-dimensional data. Metaheuristics in the other side, explore the search space systematically, looking for optimal solutions without relying on explicit problem-specific knowledge. Metaheuristics are well-suited for combinatorial optimization problems, including genome assembly as shown in the results (Table II). They can effectively handle large-scale datasets and non-linear optimization objectives. Metaheuristic algorithms offer a balance between exploration and exploitation, enabling them to escape local optima and find near-optimal solutions. However, they do not provide guarantees of finding the global optimum, and the convergence speed can vary depending on the problem and parameter settings. Researchers can combine these approaches to laverage the strengths of both and to improve genome assembly outcomes. Machine learning models can be used to guide hybrid metaheuristic algorithms combined with parallelism technologies like mapreduce in the search process like in [42] or to extract meaningful features from genomic data, enhancing the effectiveness of the optimization process of genome assembly.

VII. Conclusion

In this paper, a comprehensive review of existing literature in the field of genome assembly was established with a particular emphasis on practical algorithms. The reviewed algorithms include OLC (overlap-layout-consensus) based algorithms, de Bruijn graph-based algorithms, swarm algorithms, and machine learning methods. For each algorithm, detailed insights and highlights were provided, outlining their characteristics, strengths, and potential applications. Recent advancements in the literature also were examined, considering how these algorithms have evolved to address the challenges of genome assembly problem.In the future, the suggested DNA fragment assembler could be further developed by leveraging both machine learning techniques and nature-inspired algorithms, having the potential to be adapted into a parallel version using parallel programming frameworks like MapReduce. These enhancements are expected to lead to significantly improved performance, allowing for more efficient results and reduced execution times.

References

[1] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2002). The structure and function of DNA. In Molecular Biology of the Cell. 4th edition. Garland Science.

[2] Mountain, Andrew. ”Gene therapy: the first decade.” Trends in biotechnology 18.3 (2000): 119-128.

[3] Kalyanaraman, A. (2011). Genome Assembly. In: Padua, D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-09766-4.402

[4] Qingfeng Chen, Chaowang Lan, Liang Zhao, Jianxin Wang, Baoshan Chen, Yi-Ping Phoebe Chen, Recent advances in sequence assembly: principles and applications, Briefings in Functional Genomics, Volume 16, Issue 6, November 2017, Pages 361–378

[5] Moor J . The Dartmouth College Artificial Intelligence Conference: the next fifty years. AI Mag. 2006; 27: 87–87. doi:10.1609/aimag.v27i4.1911.

[6] Rene Y. Choi, Aaron S. Coyner, Jayashree Kalpathy-Cramer, Michael F. Chiang, J. Peter Campbell; Introduction to Machine Learning, Neural Networks, and Deep Learning. Trans. Vis. Sci. Tech. 2020;9(2):14. doi: https://doi.org/10.1167/tvst.9.2.14.

[7] ANGELERI, E., APOLLONI, B., FALCO, D. D., AND GRANDI, L. (1999). DNA FRAGMENT ASSEMBLY USING NEURAL PREDICTION TECHNIQUES. International Journal of Neural Systems, 09(06), 523–544.

[8] Constantinescu R-I. A Machine Learning Approach to DNA Shotgun Sequence Assembly. Dissertation, University of the Witwatersrand, 2015.

[9] Krachunov, M., Nisheva, M., and Vassilev, D. (2017). Machine learning models in error and variant detection in high-variation high-throughput sequencing datasets. Procedia Computer Science, 108, 1145–1154.

[10] X Huang and A Madan, CAP3: A DNA sequence assembly program, Genome Research 9 (1999), no. 9, 868–877

[11] Green,P. http://bozeman.mbt.washington.edu/ phrap.docs/phrap.html 1996.

[12] G. G. Sutton, O. White, M. D. Adams, and Ar Kerlavage, TIGR Assembler: A new tool for assembling large shotgun sequencing projects, Genome Science and Technology 1 (1995), 9–19.

[13] Idury RM, Waterman MS. A new algorithm for DNA sequence assembly. J Comput Biol. 1995 Summer;2(2):291-306. doi: 10.1089/cmb.1995.2.291. PMID: 7497130.

[14] Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci U S A. 2001 Aug 14;98(17):9748-53. doi: 10.1073/pnas.171285098. PMID: 11504945; PMCID: PMC55524.

[15] Myers, Eugene and Sutton, Granger and Delcher, Art and Dew, Ian and Fasulo, Daniel and Flanigan, Michael and Kravitz, Saul and Mobarry, Clark and Knut, Reinert and Remington, Karin and Anson, Eric and Bolanos, Randall and Chou, Hui-Hsien and Jordan, Catherine and Halpern, Aaron and Lonardi, Stefano and Beasley, Ellen and Brandon, Rhonda and Chen, Lin and Venter, J.. (2000). A Whole-Genome Assembly of Drosophila. Science (New York, N.Y.). 287. 2196-204. 10.1126/science.287.5461.2196.

[16] Batzoglou, S. (2002). ARACHNE: A Whole-Genome Shotgun Assembler. Genome Research, 12(1), 177–189. doi:10.1101/gr.208902

[17] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 2008 May;18(5):821-9. doi: 10.1101/gr.074492.107. Epub 2008 Mar 18. PMID: 18349386; PMCID: PMC2336801.

[18] Warren RL, Sutton GG, Jones SJ, Holt RA. Assembling millions of short DNA sequences using SSAKE. Bioinformatics. 2007 Feb 15;23(4):500-1. doi: 10.1093/bioinformatics/btl629. Epub 2006 Dec 8. PMID: 17158514; PMCID: PMC7109930.

[19] Jeck WR, Reinhardt JA, Baltrus DA, Hickenbotham MT, Magrini V, Mardis ER, Dangl JL, Jones CD. Extending assembly of short DNA sequences to handle error. Bioinformatics. 2007 Nov 1;23(21):2942-4. doi: 10.1093/bioinformatics/btm451. Epub 2007 Sep 24. PMID: 17893086.

[20] Dohm JC, Lottaz C, Borodina T, Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res. 2007 Nov;17(11):1697-706. doi: 10.1101/gr.6435207. Epub 2007 Oct 1. PMID: 17908823; PMCID: PMC2045152.

[21] 2010 Peng Y, Leung H, Yiu S, Chin F: IDBA-a practical iterative de Bruijn graph de novo assembler. Research in Computational Molecular Biology 2010, 426-440

[22] Bresler G, Bresler M, Tse D. Optimal assembly for high throughput shotgun sequencing. BMC Bioinformatics. 2013;14 Suppl 5(Suppl 5):S18. doi: 10.1186/1471-2105-14-S5-S18. Epub 2013 Jul 9. PMID: 23902516; PMCID: PMC3706340.

[23] Bankevich A, Bzikadze AV, Kolmogorov M, Antipov D, Pevzner PA. Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads. Nat Biotechnol. 2022 Jul;40(7):1075-1081. doi: 10.1038/s41587-022-01220-6. Epub 2022 Feb 28. PMID: 35228706.

[24] Bloom, B. H. Space/time tradeofs in hash coding with allowable errors. Commun. ACM 13, 422–426 (1970)

[25] Ye, C., Ma, Z. S., Cannon, C. H., Pop, M. and Yu, D. W. Exploiting sparseness in de novo genome assembly. BMC Bioinformatics 13, S1 (2012).

[26] Kolmogorov, M., Yuan, J., Lin, Y. and Pevzner, P. A. Assembly of long error-prone reads using repeat graphs. Nat. Biotechnol. 37, 540 (2019).

[27] Karp, R. M. and Rabin, M. O. Efficient randomized pattern-matching algorithms. IBM J. Res. Dev. 31, 249–260 (1987).

[28] Verma, Ravi and Singh, Vikas and Kumar, Sanjay. (2011). DNA Sequence Assembly using Particle Swarm Optimization. International Journal of Computer Applications. 28. 10.5120/3425-4777.

[29] K. W. Huang, J. L. Chen and C. S. Yang, "A Hybrid PSO-Based Algorithm for Solving DNA Fragment Assembly Problem," 2012 Third International Conference on Innovations in Bio-Inspired Computing and Applications, Kaohsiung, Taiwan, 2012, pp. 223-228, doi: 10.1109/IBICA.2012.8.

[30] Rajagopal, Indumathy and Sankareswaran, Uma. (2015). An Adaptive Particle Swarm Optimization Algorithm for Solving DNA Fragment Assembly Problem. Current Bioinformatics. 10. 10.2174/157489360966140301001642.

[31] Huang, KW., Chen, JL., Yang, CS. et al. A memetic particle swarm optimization algorithm for solving the DNA fragment assembly problem. Neural Comput and Applic 26, 495–506 (2015). https://doi.org/10.1007/s00521-014-1659-0

[32] Huang, Ko-Wei and Chen, Jui-Le and Yang, Chu-Sing and Tsai, Chun-Wei (2016). A memetic gravitation search algorithm for solving DNA fragment assembly problems. Journal of Intelligent and Fuzzy Systems, 30(4), 2245–2255. doi:10.3233/IFS-151994

[33] Indumathy, R and Maheswari, S Uma and Subashini, G. (2015). Nature-inspired novel Cuckoo Search Algorithm for genome sequence assembly. Sadhana. 40. 10.1007/s12046-014-0300-3.

[34] Ulker, E. D. (2016). Adaptation of harmony search algorithm for DNA fragment assembly problem. 2016 SAI Computing Conference (SAI). doi:10.1109/sai.2016.7555973

[35] Allaoui, Mohcin and Ahiod, Belaïd and El Yafrani, Mohamed. (2018). A hybrid crow search algorithm for solving the DNA fragment assembly problem. Expert Systems with Applications. 102. 10.1016/j.eswa.2018.02.018.

[36] Yassine, A., Bouzidi, M., Riffi, M.E. (2023). Cat Swarm Optimization Algorithm for DNA Fragment Assembly Problem. In: Kacprzyk, J., Ezziyyani, M., Balas, V.E. (eds) International Conference on Advanced Intelligent Systems for Sustainable Development. AI2SD 2022. Lecture Notes in Networks and Systems, vol 637. Springer, Cham. https://doi.org/10.1007/978-3-031-26384-2_57

[37] G. M. Mallén-Fullerton and G. Fernández-Anaya, "DNA fragment assembly using optimization," 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013, pp. 1570-1577, doi: 10.1109/CEC.2013.6557749.

[38] Verma, Ravi and Singh, Vikas and Kumar, Sanjay. (2011). DNA Sequence Assembly using Particle Swarm Optimization. International Journal of Computer Applications. 28. 10.5120/3425-4777.

[39] A. Ben Ali, G. Luque, and E. Alba, "An efficient discrete PSO coupled with a fast local search heuristic for the DNA fragment assembly problem," Inf. Sci., vol. 512, pp. 880–908, Feb. 2020, doi: 10.1016/j.ins.2019.10.026.

[40] Jain, Sehej and Bharti, Kusum. (2021). Chaos inspired Particle Swarm Optimization with Levy Flight for Genome Sequence Assembly.

[41] Karthik, GVSK and Deb, Sankha. (2017). A Methodology for Assembly Sequence Optimization by Hybrid Cuckoo-Search Genetic Algorithm. Journal of Advanced Manufacturing Systems. 17. 10.1142/S021968671850004X.

[42] Raja, G., and Reddy, U. S. (2017). Nature Inspired Algorithms for Genome Subsequence Assembly in Hadoop. 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).

[43] Churchill, G., Burks, C., Eggert, M., Engle, M., and Waterman, M. (1993). Assembling DNA sequence fragments by shuffling and simulated annealing. Technical Report LAUR 93-2287, Los Alamos National Lab., Los Alamos, NM

[44] C. Burks, M. Engle, S. Forrest, R. Parsons, C. Soderlund, P. Stolorz, Stochastic optimization tools for genomic sequence assembly, in: M. Adams, C. Fields, J. Venter (Eds.), Automated DNA Sequencing and Analysis, Academic Press, 1994, pp. 249–259.

[45] Burks, C., Engle, M., Lowenstein, M., Parsons, R., and Soderlund, C. (1993). Stochastic optimization tools for DNA assembly: integration of physical map and sequence data. Poster presented at Genome Sequencing and Analysis Conference V.

[46] Parsons, Rebecca J.; Forrest, Stephanie; Burks, Christian (1995). Genetic algorithms, operators, and DNA fragment assembly. Machine Learning, 21(1-2), 11–33. doi:10.1007/bf00993377

[47] Alba, E., Luque, G., and Khuri, S. (n.d.). Assembling DNA Fragments with Parallel Algorithms. 2005 IEEE Congress on Evolutionary Computation. doi:10.1109/cec.2005.1554667

[48] Saidi, Naima and Abdesslem Layeb. "A hybrid Chemical Reaction Optimisation Algorithm For Solving The DNA Fragment Assembly Problem." Conference on Innovative Trends in Computer Science (2019).

[49] Alba, E., and Luque, G. (2007). A New Local Search Algorithm for the DNA Fragment Assembly Problem. Lecture Notes in Computer Science, 1–12. doi:10.1007/978-3-540-71615-0-1

[50] Abdelkamel Ben Ali,Gabriel Luque,Enrique Alba,Kamal E. Melkemi (2017). An improved problem aware local search algorithm for the DNA fragment assembly problem. Soft Computing, 21(7), 1709–1720. doi:10.1007/s00500-015-1875-2

[51] Mohamed Abdel-Basset, Reda Mohamed, Karam Sallam, Ripon Chakrabortty, and Mike Ryan. An efficient-assembler whale optimization algorithm for dna fragment assembly problem: Analysis and validations. IEEE Access, 8:222144–222167, 01 2020.

[52] S. Mirjalili and A. Lewis,The whale optimization algorithm, Adv.Eng. Softw., vol. 95, pp. 51 67, May 2016.

[53] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, MI: The University of Michigan Press

[54] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Publishing Company.

[55] Forrest, S. (1993). Genetic algorithms: Principles of natural selection applied to computation. Science 261:872-878.

[56] Li, Ching. "DNA Fragment Assembly Algorithms: Toward a Solution for Long Repeats." (2008).

[57] Alba, E., and Luque, G. (2008). A Hybrid Genetic Algorithm for the DNA Fragment Assembly Problem. Studies in Computational Intelligence, 101–112. doi:10.1007/978-3-540-70807-0-7

[58] J.A. Hughes, S. Houghten, D. Ashlock, Restarting and recentering genetic algorithm variations for DNA fragment assembly: The necessity of a multi-strategy approach, Biosystems 150 (2016) 35–45.

[59] Uzma, ; Halim, Zahid (2020). Optimizing the DNA fragment assembly using metaheuristic-based overlap layout consensus approach. Applied Soft Computing, 92(), 106256–. doi:10.1016/j.asoc.2020.106256

[60] Bennaceur, Hachemi and Almutairy, Meznah and Alqhtani, Nora. (2020). An Investigative Study of Genetic Algorithms to Solve the DNA Assembly Optimization Problem. International Journal of Advanced Computer Science and Applications. 11. 10.14569/IJACSA.2020.0111019.

[61] Bennaceur, Hachemi and Almutairy, Meznah and Alqhtani, Nora. (2023). Experimental Evaluation of Genetic Algorithms to Solve the DNA Assembly Optimization Problem. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.0140333.

[62] Meksangsouy, P., and Chaiyaratana, N. (n.d.). DNA fragment assembly using an ant colony system algorithm. The 2003 Congress on Evolutionary Computation, 2003. CEC '03. doi:10.1109/cec.2003.1299885

[63] Wetcharaporn, Wannasak and Chaiyaratana, Nachol and Tongsima, Sissades. (2006). DNA Fragment Assembly by Ant Colony and Nearest Neighbour Heuristics. 1008-1017. doi: 10.1007/11785231.106.

[64] Wetcharaporn, Wannasak and Chaiyaratana, Nachol and Tongsima, Sissades. (2006). DNA Fragment Assembly: An Ant Colony System Approach. 231-242. 10.1007/11732242_21.

[65] Firoz, Jesun and Rahman, Mohammad and Saha, Tanay. (2012). Bee Algorithms for Solving DNA Fragment Assembly Problem with Noisy and Noiseless data. GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation. 10.1145/2330163.2330192.

[66] Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. BioInspired Comput. 2(2), pp. 78–84 (Mar 2010)

[67] Vidal, Pablo and Olivera, Ana. (2018). Solving the DNA fragment assembly problem with a parallel discrete firefly algorithm implemented on GPU. Computer Science and Information Systems. 15. 9-9. 10.2298/CSIS170510009V.

[68] Mallén-Fullerton, G. M. , Hughes, J. A. , Houghten, S. , and Fernández-Anaya, G. (2013). Benchmark datasets for the dna fragment assembly problem. International Journal of Bio-Inspired Computation, 5 (6), 384–394 .

[69] Formenti, Giulio, and Kerstin Howe. "An assembly line for an improved human reference genome." NATURE 611.7936 (2022).