# Optimizing the DNA Fragment Assembly using Metaheuristic-based Overlap Layout Consensus Approach

Uzma[*] and Zahid Halim

**Abstract**
Nucleotide sequencing finds the exact order of nucleotides present in a DNA molecule. The correct DNA sequence is required to obtain the desired information about the complete genetic makeup of an organism. The DNA fragment assembly correctly combines the DNA information present in the form of fragments as a sequence. Reconstruction of the original DNA sequence from large fragments is a challenging task due to the limitations of the available technologies that reads the DNA sequence. Objective of the DNA fragment assembly is to find the correct order of the fragments which is further used in the generation of a consensus sequence that represents the original DNA sequence. Power Aware Local Search (PALS) algorithm proposed for the DNA fragment assembly is an efficient method that orders the fragments in a correct sequence by minimizing the number of contigs. This work presents a hybrid approach on the basis of Overlap Layout Consensus for the DNA fragment assembly, where Restarting and Recentering Genetic Algorithm (RRGA) with integrated PALS is utilized as an evolutionary operator. Quality of the current proposal is quantified using overlap scores and the number of contigs. This work is evaluated using 25 benchmark datasets with three types of experiments. The results are compared with four state-of-the-art methods for the same task, namely, Recentering-Restarting Genetic Algorithm variation for DNA fragment assembly, PALS, Genetic Algorithm, and Hybrid Genetic Algorithm. Results show better average performance of the proposed solution.

**Keywords**: Metaheuristic, DNA fragment assembly, Hybrid genetic algorithm, Overlap layout consensus, Optimization

## 1. Introduction

Deoxyribonucleic acid (DNA) is a complex organo-chemical molecular structure, acting as a hereditary material. It is responsible for the storage and flow of genetic information present in all eukaryotes and prokaryotes including some viruses. It is a double stranded thread like helical structure positioned inside the nucleus of a cell. The two strands of DNA are wrapped around each other via chemical interactions such as Hydrogen bonding and Phosphodiester linkages between complementary bases and nucleotides, respectively [1]. Each strand comprises of a long stretch of nucleotides which consist of sugar without Oxygen atom at position 5 (deoxyribose sugar). This is linked to a phosphate group and one of the possibly four nucleotides which may be purines (Adenine & Guanine- A, G) or pyrimidine (Cytosine and Thymine-C, T). However, the binding of these two strands is base specific, i.e., adenine will only bond with thymine and cytosine will only interact with guanine. The DNA has coding and non-coding regions termed as exons and introns, respectively. Coding regions of the DNA are known as genes. The genetic information is stored in the form of chemical bases that are termed as nucleotides. This information is transcripted into Messenger ribonucleic acid (mRNA) which is later translated into a protein. This protein performs an encoding function, such as structural building or acting as a signaling molecule. The normal life cycle goes on in this way [2]. Alterations in the sequence of nucleotides do happen every single day because of the genetic or environmental fluctuations. These are called mutations that may either cause a variation or develop into a disease. An example disease is cancer if not repaired by the built-in mismatch repair mechanisms. These circumstances occur because the gene product, that is typically a protein, is sequence specific and proteins differ from each other on the bases of their amino acid sequence in the form of triplets (codons) [3]. Therefore, a nonsynonymous single nucleotide polymorphism (nsSNP) or missense mutation in the genomic sequence results in the translation of incorrect amino acid sequence. This yields an abnormal protein (mass) production or causes failure of the required protein production machinery that distracts the normal cell cycle mechanism. All of this develops into various biological syndromes. The chromosome is a protein-DNA complex on which the long DNA molecule is packaged acting as a template for the newly synthesized DNA molecules during cell division. The DNA has stable configuration as compared to RNA and is negatively charged molecule as a consequence of present phosphate ions that make-up the double-helical structure of the DNA.

With the introduction of biomedical and bioinformatics tools along with the advancement in molecular biology, a gene or DNA as a whole, can be sequenced for numerous purposes. However, there is still a long way to go in order to

[*] Corresponding author

*Uzma and Z. Halim are with the Machine Intelligence Research Group (MInG), Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan, 23460. E-mail: {uzma, zahid.halim}@giki.edu.pk.*

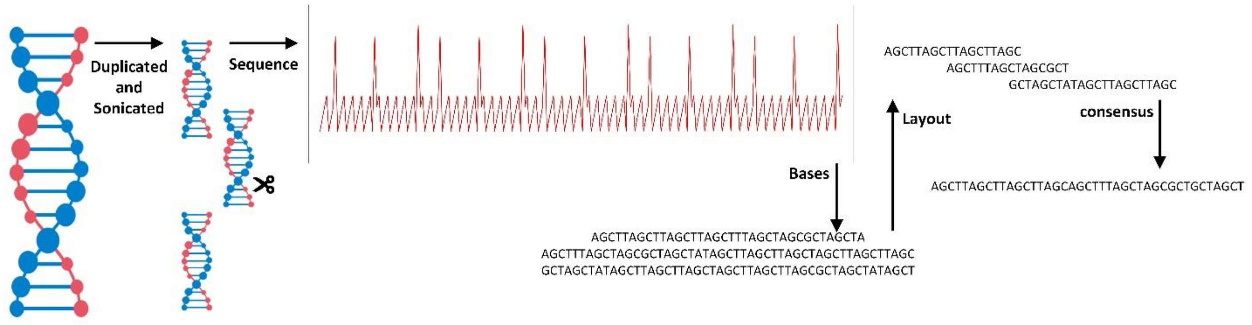

Duplicated and Sonicated
Sequence
Bases
AGCTTAGCTTAGCTTAGCTTTAGCTAGCGCTAGCTA
AGCTTTAGCTAGCGCTAGCTATAGCTTAGCTTAGCTAGCTTAGCTTAGC
GCTAGCTATAGCTTAGCTTAGCTAGCTTAGCTTAGCGCTAGCTATAGCT
Layout
AGCTTAGCTTAGCTTAGC
AGCTTTAGCTAGCGCT
GCTAGCTATAGCTTAGCTTAGC
consensus
AGCTTAGCTTAGCTTAGCAGCTTTAGCTAGCGCTGCTAGCT

**Fig. 1.** The phases of DNA sequencing and assembly.

unveil the DNA completely and comprehend the mechanisms responsible for various biological syndromes, focusing on the hazy image of biological phenomenon. Fig. 1 shows a typical DNA structure, DNA sequencing, and assembly phases.

The technique practiced to determine the exact order of nucleotide sequence within a DNA molecule is termed as DNA sequencing. The DNA molecule is a long stretch of nucleotides in a particular manner that defines the complete life of an individual. Therefore, it is imperative to recognize the exact sequence of this molecule to comprehend the biological phenomenon and face the future challenges. However, sequencing the entire genomic DNA of an organism is a complex and laborious task. Therefore, to sequence the long DNA stretch, it is chopped down into smaller fragments and the smaller sequences are assembled together to get a consensus sequence. Various techniques are used for DNA sequencing and each has its own pros and cons. The classical sequencing methods include sanger sequencing, Maxam and Gilbert sequencing, and Next Generation Sequencing (NGS) that include Ion-Torrent:Proton, Roche 454 and Illumina (Solexa) sequencing [4], to name a few. Sanger sequencing is an enzymatic way of sequencing, which is also known as classical or the first generation sequencing. Its workflow is designed by considering the mechanism of nucleotides addition to form a DNA sequence. The query DNA is extracted and purified via in-vivo techniques. Next generation sequencing is a modern day sequencing technique because of its high throughput that include different technologies such as Ion-Torrent: Proton Sequencing, Roche 454 and Illumina (Solexa) sequencing. Though these technologies are executed and supported by various companies, however, they all have a common principle that involves amplicons reading. The advancement and optimization in sequencing technologies are reducing the cost with high throughput. Thus, the whole genome of an individual can be sequenced nowadays for a few thousand dollars. Despite these improvements, robust statistical and bioinformatics approaches are required to translate the genomic data into a meaningful form which can be analyzed and hence, it becomes vital to develop informatics algorithms to understand the true meaning of the data. The soft computing methods can play a vital role in this context.

### 1.1. DNA fragment assembly

DNA fragment assembly is a method of merging small fragments of DNA sequences into longer orders to reconstruct the original sequence. This long sequence, as a result of fragment assembly, is termed as 'contig', a sequence whose validity can be inspected via alignment with the reference genome. There are thousands of genes present on a human DNA, however, only a small percentage of them have been discovered so far [5]. Obtaining the exact DNA sequence is an important undertaking to find the exact location of these genes. The current technologies available for this task have limitations in sequencing the genome as a whole (in a single step). The human DNA has 3.2 billion nucleotides. However, even a read of more than 600 base pairs (bp) cannot be sequenced with certainty. Therefore, it is necessary to divide, the DNA into short pieces which can be sequenced conveniently with confidence. This procedure is called the shotgun sequencing. In this approach, each copy of the DNA portion (which is huge in numbers) is broken down into smaller fragments that can be automatically read by the sequencer. This results in the problem of orphan fragments as this process does not maintain details about the fragments' order or knowledge about the portion from where these fragments came. This leads to the DNA fragment assembly problem where the overlapping locations are defined as landmarks to reassemble them for reconstruction of the original DNA sequence.

Various methods are available for DNA fragment assembly. The most common is the Overlap Layout Consensus (OLC) approach [6]. This involves three steps, construction of an overlap graph which is followed by a layout step in which stretches of the overlapping graphs are bundled into contigs and then finally the most likely nucleotide for each contig is chosen in the consensus step. Suffix trees or dynamic programming is employed to find the overlapping regions in which reads' length, total length, number of overlapping reads pair/global alignment recurrence, and score function are taken into consideration. Although, OLC paradigm has been followed by fragment assembly in DNA sequencing for more than 30 years, however, it is still recommended to employ de Bruijin graph (dBG) as the repeating regions are better assembled by it in comparison to the OLC. With the passage of time, advancements in NGS technology has been able to produce comparatively longer reads. This allows the OLC technique to be promising [7]. However, neither OLC nor dBG is static. Issues are raised due to these methods in genomic shotgun sequencing, but they are suitable for cloning assemblies. Pevzner et al. [8] presents an approach that bypasses the "repeat issues" which arises from the excessive number of contigs that makes the finishing unnecessarily complicated as they require screening of the whole genome even for a single error.

Multiple strategies have been designed for the DNA fragment assembly problem in the past [9]. These strategies involve metaheuristic and heuristic methods in addition to machine learning and computational intelligence approaches. Some of the soft computing techniques used for DNA fragment assembly include Ant Colony Optimization (ACO) [10], Particle Swarm Optimization (PSO) [11], greedy algorithms [12] and structured pattern matching. This problem domain customizes the existing algorithms to improve the assessment of entire genome. Sanger sequencing or chain termination method [13] produces a high-quality sequence for comparatively longer stretches of DNA (900 bp long). It is commonly used for sequencing singular pieces of the DNA. However, for the entire genome sequencing, the Sanger technique is not appropriate because of it being expensive.

## 1.2. Motivation and problem statement

The key motivation of this work is to utilize the soft computing methods to device a framework that can solve the DNA fragment assembly problem. This work aims to develop a solution based on the Evolutionary Algorithms (EAs) from the domain of soft computing. The choice of EAs is made based on their proven quality of results [14] and the ability to search an optimum solution in large search spaces [15]. The proposed work adds to the existing pool of solutions for the DNA fragment assembly problem. Where, the end user could opt for the present proposal based on the features of the particular dataset at hand.

As mentioned above, various technologies and algorithms have been designed in the recent past for the sequencing of nucleotide strands. However, the task of sequencing the strand (having the bp length of more than 600) with higher confidence is yet an open research problem. In the shotgun sequencing technique, the DNA is replicated many times via Polymerase Chain Reaction (PCR) and later each amplified fragment is divided into many small cut pieces. This process has its benefits, however, it does not keep track of the fragments' order. This leads to the problem of fragment assembly. The objective of DNA fragment assembly is to find the correct order of fragments which is further used for the phase of consensus sequence representing the original DNA sequence. The approach in [16] combines heuristics with RRGA to improve its validity. This method uses the overlap score (as a fitness value) between the adjacent fragments for testing the quality of DNA fragment assembly. Power Aware Local Search (PALS) algorithm proposed for DNA fragment assembly is an efficient technique. A hybrid approach is defined in the present proposal, where the RRGA [17] use PALS as an evolutionary operator. The work in [16] uses the classical measure such as the overlap score between the adjacent fragments for the judgment of the assembly quality. However, the main objective of DNA fragment assembly is to find the order of the fragments which can minimize the number of contigs. The quantification of contigs estimation ensures the fragments order based on cutoff. The main objective here is to minimize the number of contigs in addition to maximizing the fitness value of a proposed set of fragments for the DNA fragment assembly problem (DNA-FAP). The solution obtained through two objectives (reduced contigs number and maximum fitness value) best represents the DNA sequence assembly.

However, the issue remains that one cannot always prefer a high overlap score because sometimes, a strand having a higher number of contigs also has the highest number of overlap scores. Therefore, to judge the actual quality of the DNA fragment assembly, this work utilizes PALS to calculate the number of contigs with RRGA to enhance the DNA fragment assembly quality. Formally, the problem statement of this work is:

*To order the fragments while minimizing the number of contigs and maximize the overlap score. This is to be accomplished by avoiding the local optima to improve the quality of DNA fragment assembly (DFA).*

## 1.3. Key contributions and novelty

This work focuses on the DNA-FAP. It is concerned with the reconstruction of the target DNA. For this, it identifies the right order and orientation of nucleotides for each fragment in the layout by taking a vast number of sequenced fragments into consideration. Usually, when the reads are generated from a sequencing technology, it does not keep track of the fragments order leading to the problem of DNA fragment assembly. This work presents a solution to optimize the DFA.

The past work on DFA has used the sum of overlap scores for the judgment of assembly. However, the sum of overlap scores do not prove that the reads are in their right order. This is because there is a possibility that some reads are adjacent based on higher overlap scores while others have low score. Thus, there are chances that a strand having maximum sum of overlap scores will produce large number of contigs. Therefore, the current work introduces a novel evolutionary technique that uses the local search algorithm as a genetic operator. This creates a strand having reads in order while minimizing the number of contigs by setting a threshold between adjacent fragments and having maximum sum of overlap scores.

The current proposal presents a bi-objective optimization problem addressed through two main objectives, i.e., (a) maximizing the sum of overlap score, and (b) minimizing the number of contigs. For this, the proposed work focuses on OLC approach. For initializing the fragment positions, the Traveling Salesman Problem (TSP) approach is used. It is acknowledged that TSP finds the optimum solution efficiently in all benchmark datasets used here. The TSP heuristic, i.e., 2-opt is used to reset position of the fragments. Recentering-Restarting Hybrid Genetic Algorithm (RRHGA) is used to explore all feasible solutions. Due to its sequential search capability, RRHGA avoids premature convergence and thus orders the fragments while maximizing the overlap scores and minimizing the number of contigs. For performance evaluation, 25 benchmark datasets including the collection of 14 f-series datasets are used in this study. Three sets of experiments are conducted using these datasets by utilizing two approaches; (i) with force recentre and (ii) without force recentre. The solution obtained via RRHGA is further evaluated by the number of contigs and sum of overlap scores. As a result of this implementation, the final reads are better. This proposal addresses the issues faced by reconstructing the original DNA template from the fragments using soft computing methods.

### 1.4. Paper organization

The rest of the paper is organized as follows. Section 2 presents the related work on DNA fragment assembly. This section also lists synthesis of the state-of-the-art and separately mentions the limitations of the past contributions that the present proposal addresses. Section 3 explains the biological process and material utilized in this work including the Sanger sequencing, next generation sequencing, and the 25 benchmark datasets used during experiments. Section 4 presents the proposed solution. This section first lists detail on existing methods that are incorporated in the proposed solution followed by the core of the proposed method. Section 5 lists the conducted experiments and obtained results. Section 6 discusses the results and performance of the competing methods. Finally, Section 7 concludes this work and mentions a few of the future directions.

## 2. Related work

This section highlights the previously reported works in the domain of DNA fragment assembly. The section also covers the current efforts to optimize available techniques along with the broad view and current state of DNA-DFA related challenges.

### 2.1. Past contributions

DNA fragment assembly is a strategy that endeavors to reconstruct the original DNA sequence from a large number of fragments that are several hundred bp long. Optimization of the DNA fragment assembly is crucial due to the limitations of presently available sequencing technologies that must be considered to address the challenges in the field of genome biology. For instance, the human DNA is around 3.2 billion nucleotides long and cannot be read at once. Typically, the DNA molecule is sliced randomly at first to get many shorter fragments that can be sequenced easily. The original DNA is reconstructed from overlapping the DNA fragments, called shotgun sequencing strategy. Initially, the assembly of short fragments was performed manually. To automate the shotgun arrangement assembly, abundant resources and efforts have been dedicated to explore new methods and approaches. Numerous literary works present solutions for DNA sequence assembly issue. The general framework of most assembly algorithms is to first find the overlap scores by comparing all pairs of fragments, taken after framing an inexact layout of fragments. Afterwards, they make a consensus sequence. The general framework of most assembly algorithms is to first arrange competitor covers by inspecting all set, trailed by shaping a surmised format of parts. Finally, making an accord succession. The fragment assembly is an non-deterministic polynomial-time hard (NP-hard) problem, which is unsolvable in polynomial time. There are three separate stages that divides the DNA fragment assembly more specifically into three phases as shown in Fig 1. These include: overlap phase, layout phase, and consensus phase.

The overlap phase finds the overlapping fragments. This phase find fragments having the largest similarity between the suffix of one sequence and prefix of the other. Usually a dynamic programming algorithm is implemented in finding the similarity between fragments. In the layout phase, fragments are ordered based on a similarity index, which is the main objective of this phase. An alignment algorithm is applied to the correct order of fragments that combines the whole pairwise alignments obtained in the overlap phase. This becomes a bit more challenging due to unknown orientation, repeated regions, base call errors, incomplete coverage, and chimeras/contamination. Finally, to derive the DNA sequence from the layout phase, the most common technique used is the majority rule in building the consensus.

Ignatov et al. [18] present a fragmentation technique named Fragmentation Through Polymerization (FTP). Their method produces double-stranded DNA fragments. These fragments are directly usable for the NGS library construction. The advantage of their technique is that it eliminates the additional step of DNA end reparation. In their work Lorenz curves are utilized to access the read coverage for its uniformity along the entire genome. The FTP method enable to reduce the processing time from 180 minutes to 110 minutes. This ultimately will enable to reduce the cost of library creation. The work in [19] presents an approach based on the quantum-inspired genetic algorithm [20, 21] for the DNA fragment assembly. Their method is primarily focused to perform the de novo assembly of DNA fragments. This is done using the OLC approach. Authors have compared their method with three approaches for solving DFA problem, namely, Genetic Algorithm (GA), PSO, and ACO-based metaheuristic. Comparison is made based on the overlap score and number of contigs. Richter et al. [22] present a DNA-assembly system named as Zero-Background Redα (ZeBRα), for cloning multiple DNA-fragments. According to the authors, their method reduces both time and cost. Their method addresses the issues in routine and high-throughput cloning. Raja et al. [23] introduce a method named MapReduce Maximum Exact Matches (MR-MEM) for searching and then mapping genome subsequences. The key contribution of their work is the utilization of MapReduce, a parallel execution framework. Alignment in their work is done based on a reference genome. For this, a fragment subsequence is initially matched with the genome to recognize the probable matching sites.

Frequently used methods for DFA are naturally inspired algorithms such as ACO, PSO, and GA. Firoz et al. [24] present a hybrid metaheuristic that use Artificial Bee Colony (ABC) algorithm with PALS. Their method finds a permutation solution based on two objective functions; maximizing the overlap score and minimizing the number of contigs for choosing the best solutions. Hughes et al. [16] propose the OLC-based approach for DNA fragment assembly. Their algorithm merges two methods; heuristic and computational intelligence. The combination of GA variations such as island model, recentering restarting, and ring species are used to resolve the DNA-FAP. Heuristic algorithms such as 2-opt and the Lin-Kernighan [25] also employ the combination of GA. These combinations take advantage~~s~~ of heuristic and computational intelligence while suppressing existing shortcomings. Results show high quality performance when heuristic is used as an initial seed. Heuristic functions are useful in approximating exact values. They produce results within a shorter period that are still significant; however, they may be reduced to local optima. To overcome this issue, a more precise algorithm such as Recentering-Restarting Genetic Algorithm (RRGA) can be employed, which is devised to avoid local optima, thus improving the results. The strengths of this method include easy utilization, and the ability to avoid local optima. Result quality can improve further when RRGA is used with heuristic and a variation of the GA. The GA variations perform well as compared to the results of the previous algorithms for DNA-FAP.

Alba et al. [26] propose a heuristic called PALS for DNA fragment assembly: a variation of Lin's 2-opt. Their designed approach not only finds the scored overlaps, but also calculates the number of contigs that are created or destroyed when the tentative solution is manipulated. Traditional assembler finds the best solution having strong overlap scores between the adjacent fragments in the layout. However, the goal of DNA fragment assembly is to minimize the number of contigs using a specific order of the fragments. In some cases they report that the sum of all the overlap scores (fitness value) of a solution is better than the one generated by many contigs. This suggest that the fitness ought to be supplemented with the real number of contigs. The comparisons of their approach is made with commercially available GA packages, e.g., Phrap and a pattern matching algorithm. The comparison is made based on the final number of contigs. Where, PALS perform better or similar to the techniques utilized for comparison.

Minetti et al. [27] set out to analyze the DNA-FAP for noisy data [28]. DFA is one of the important tasks of the genome project. The project depends on DFA accuracy and efficiency. The methods used for sequencing large DNA

**Table 1**
Key features of the past works and current proposal.

| Work | Fitness value: Utilize overlap score? | Utilize no. of contigs | 2-opt for initial solution | Computational intelligent method | PALS as evolutionary operator | Premature convergence |
|---|---|---|---|---|---|---|
| Hughes et al. [16] | ✓ | ● | ✓ | RRGA | ● | Avoids premature convergence |
| Alba et al. [26] | ✓ | ✓ | ● | Hybrid Genetic Algorithm | ✓ | Premature convergence |
| Alba et al. [43] | ✓ | ✓ | ● | Local Search Algorithm | ● | Traps in local optima |
| Rathee et al. [19] | ✓ | ✓ | ● | Quantum-inspired GA | ● | May occure |
| Firoz et al. [24] | ✓ | ✓ | ● | ABC + GA | ● | May occure |
| Parsons et al. [42] | ✓ | ● | ● | GA | ● | Premature convergence |
| Current work | ✓ | ✓ | ✓ | RRHGA | ✓ | Avoids local optima and premature convergence by using RRGA |

strands in laboratory produces noisy data. Therefore, the data used for assembling the fragments contain errors. The analysis in [27] is completed using three well-known algorithms, namely, GA, PALS, and simulated annealing for the DFA problem in the case of noiseless and noisy instances. Their behaviors were observed for each scenario at various stages of the algorithm~~s~~. For both scenarios, the performance is evaluated by executing each instance 30 times. While comparing the algorithms for noisy instance, experimental data show GA performance to be better than PALS and SA. However, for noiseless instances, the SA performs better than the other two.

Kikuchi et al. [29] propose an approach based on GA for DNA-FAP. Their algorithm is based on two parts: the first part is about shortening the chromosome length called a Chromosome Reduction Step (CRed). Whereas, the second part is about greedy heuristic called Chromosome refinement step (CRef), which is implemented to enhance local chromosome fitness. The algorithm combines longer contigs with shorter gaps between them. The algorithm is designed such that the user can pause it at any moment to check whether the results meet their criteria. They use artificially created genomes to carry out their experiments. First, a genome of 1000 bp is randomly constructed. Later, 100 fragments are generated from the genome. From these 100 fragments, the chromosomes of the GA are arranged randomly. The probability of crossover and mutation is set at 0.5 and 0.005, respectively. After 30000 generations, the algorithm succeeds in finding sequence covering 98% of the original DNA.

Hughes et al. [30] present a hybrid algorithm that combines heuristic and GA variations for DNA fragment assembly. They use the overlap layout consensus approach among a variety of approaches, such as a de Bruijn graph and greedy algorithm. The study of RRGA with different variations such as RRGA+Ring Species and RRGA+island model produces quality results with direct/indirect representation. The experiments are seeded with two different solutions; random seed and a local search algorithm for TSP called 2-opt. The solution seeded with 2-opt produce better result than the random seed approach. Lin-Kernighan algorithm is also used as a starting seed. The results produced in their work show RRGA and GA variations to yield high-quality results. Combination of the heuristic and GA variation produces quality results on sixteen benchmark datasets in their proposal.

Mallén-Fullerton [31] propose an algorithm based on PSO and Differential Evolution (DE). The experiments are executed in two parts. Firstly, the overlap matrix is calculated and insignificant fragments are excluded. Secondly, the Lin Kernighan algorithm is used for the TSP problem and then the Concorde program is implemented for verification of the Lin Kernighan algorithm. Their approach applies TSP variation and achieves global optima for all benchmark datasets. They present a collection of benchmark datasets used for DNA fragment assembly. Their work presents an accumulation of benchmark datasets[1] for an extensive variety of fragment lengths, sequence lengths, and a number of fragments alongside portrayal of the technique used to create them.

Hughes et al. [17] develop multiple variations of recentering-restarting evolutionary algorithms. This examination builds different varieties of this algorithm with the end goal of assessing its utilization for ordered-gene problems. Two distinctive versatile representations are investigated that create sets to deliver local search operations. Their approach has the ability to change the number of transpositions to adjust the measure of the search space. Hughes et al. also present the RRGA as an approach for improving the heuristic results. Two experiments of test analysis are performed. The first one utilizes large problem instances to analyze how well the algorithm performs in contrast to

[1] http://chac.sis.uia.mx/fragbench/

the benchmarks from the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) TSP usage challenge. The second one utilizes numerous smaller problem samples to evaluate performance of the RRGA. Their results demonstrate that the RRGA can produce noteworthy outcomes over a fundamental GA when applied to the TSP variant.

Meksangsouy [1] propose an asymmetric ordering representation. The search solution in their work is represented by a colony produced by the ants' created paths. For each pair of successive fragments, the overlap scores are summed to evaluate the optimality of the layout. Two types of experiments are performed: single-contig and multiple-contig problems. The experiments for single contig problem indicate that the results for ant colony system algorithm and nearest neighbours heuristic are approximately the same. However, considering multiple-contig problems, ant colony algorithm performs better than the nearest neighbours heuristic. Nebro et al. [32] propose a GA in the computational grid for DNA fragment assembly. It uses panmictic population and computes several individual evaluations in parallel.

## 2.2. Synthesis of the state-of-the-art

Ignatov et al. [18] presented the FTP method that produces double-stranded DNA fragments. An advantage of their technique is that it eliminates the additional step of DNA end reparation. Their method has the limitation of producing a large number of dangling reads. However, in comparison to a commercial enzymatic techniques, i.e., fragmentase, the difference is less than 1%. The quantum-inspired GA for the DNA fragment assembly in [19] focus on de novo assembly utilizing OLC approach. It has the ability to find the solution in a large search space. However, there is a possibility that the GA-based solution may stuck in local optima.

The proposal by Firoz et al. [24] present ABC-FAP integrated with PALS. They have the advantage of utilizing two objectives, i.e., maximizing the overlap score and minimizing the number of contigs for choosing the best solutions. The ABC-FAP finds a permutation solution based on the two objective functions. They use the bee algorithm for the task which has the limitation of requiring a new fitness tests on new algorithm parameters. Additionally, its processing speed is slow when executed sequentially. The solution in [16] propose the OLC-based approach for DNA fragment assembly utilizing GA. Like the proposal in [19] there is a possibility that the GA-based solution may stuck in local optima. Hughes et al. [17] develop multiple variations of recentering-restarting evolutionary algorithms. Their method has been used in the DFA domain. Despite of being an evolutionary approach, it has the capacity to escape the local optima. However, this is at the cost of additional computations and execution time. Nebro et al. [32] propose a GA in the computational grid for DNA fragment assembly. Utility of the grid has its advantage of lower access time. However, their work needs to be evaluated on large and more complex instances of the DNA-FAP as well.

## 2.3. Limitations of the past works addressed in the present proposal

The past works, like [16] and [26] proposed hybrid algorithm for DNA fragment assembly to enhance its validity. These methods utilized GA which causes premature convergence. The present proposal combines the heuristics with the computational intelligence approach. Here, the overlap score is utilized as a fitness value for the judgment of an assembly quality which is the main limitation of past works. Similarly, other works in the past designed a local search algorithm that minimizes the number of contigs to ensure the correct order of the fragments. However, the limitation of local search algorithm utilization is that it cannot be further improved and gets trapped in a local optimum. Hence, the shortcoming in the existing work is that one cannot focus on high overlap score alone because sometimes a strand having maximum number of contigs will also have the highest overlap score. The main objective of DNA fragment assembly is to correctly order the fragments while minimizing the number of contigs with the maximum overlap score. The solution obtained using these two objectives in the present proposal best represents the DNA fragment assembly. Therefore, this work designs a hybrid algorithm that combines RRGA with PALS. This approach orders the fragments while minimizing the number of contigs and maximize the overlap score. This is done here while avoiding the local optima to improve the quality of DFA. Table 1 lists the key features of the current proposal and closely related past works.

**Table 2**
Datasets detail.

| Genre | Benchmark[2] | Coverage | Mean fragment length | Number of fragments | Original sequence length |
|---|---|---|---|---|---|
| GenFrag instances | x60189 5 | 5 | 286 | 48 | 3,835 |
| | x60189 6 | 6 | 343 | 66 | |
| | x60189 7 | 7 | 387 | 68 | |
| | m15421 5 | 5 | 398 | 398 | 10,089 |
| | m15421 6 | 6 | 350 | 350 | |
| | m15421 7 | 7 | 383 | 383 | |
| | j02459 7 | 7 | 405 | 352 | 20,000 |
| | bx842596 4 | 4 | 708 | 442 | 77,292 |
| | bx842596 7 | 7 | 703 | 773 | |
| DNAgen instances | acin1 | 26 | 182 | 307 | 2,170 |
| f-series | f25_305 | 25 | 307 | | |
| | f25_400 | 25 | 400 | | |
| | f25_500 | 27 | 500 | | |
| | f50_315 | 50 | 315 | | |
| | f50_412 | 50 | 412 | | |
| | f50_498 | 50 | 498 | | |
| | f100_307 | 100 | 307 | | |
| | f100_415 | 100 | 415 | | |
| | f100_512 | 100 | 512 | | |
| | f508_354 | 508 | 354 | | |
| | f635_350 | 635 | 350 | | |
| | f737_355 | 737 | 355 | | |
| | f1343_354 | 1343 | 354 | | |
| | f1577_354 | 1577 | 354 | | |

## 3. Biological process and material

This section presents the related biological concepts and material used in this work. The section starts with a brief introduction to Sanger sequencing that is followed by the details about NGS. Additionally, key features of the 25 benchmark datasets used here for analysis of the proposed algorithm are listed.

### 3.1. Sanger sequencing

Frederick Sanger [33] introduced an enzymatic way of sequencing, which is also known as classical or the first generation sequencing. The workflow is designed by considering the mechanism of nucleotides addition to form a DNA sequence. The query DNA is extracted and purified via in-vivo techniques. Four parallel PCR reactions are executed. These PCR reactions differ from the normal PCR. In case of Sanger sequencing, as the raw material includes an additional supplement, termed as dideoxynucleotides (ddNTPs) along with dNTPs, obstructs the nucleotide chain formation upon their addition. The ddNTPs lacks OH at position three which hampers the attachment of the next nucleotide during chain formation and thus it limits the fragment length. The purpose of adding these specialized ddNTPs is to make copies of the template strand with varying lengths. These fragments are then run on a poly acrylamide gel in order to get a sequence up to a few hundred bases. This is also known as chain termination technique which is further optimized by tagging a luminous dye (sparkling color) to each particular nucleotide. This makes it possible to run all the four reactions in a single go. The classical acrylamide gel is replaced by polyacrylamide gel in order to have better execution time. Low throughput makes this technique non-scalable as it is not a multiplex approach.

[2] http://chac.sis.uia.mx/fragbench/

## 3.2. Next generation sequencing

The second (next) generation sequencing is a modern day sequencing technique because of its high throughput that include different technologies such as SOLid, Ion-Torrent: Proton sequencing, Roche 454, and Illumina (Solexa) sequencing. Though these technologies are executed and supported by various companies, however, they all have a common principle that involves amplicons reading. It is spatially separated followed by a clonal amplification in a cyclic parallel pattern. The NGS has multiplex property. Referring to the technology, a couple of steps are there in DNA sequencing. Initially, the sample of DNA is shattered into pieces of particular size depending on the sequencing approach to be utilized. Mechanical techniques are used to shatter the DNA into fragments such as Sonication. This process can also be done via enzymatic activity which makes the process easier to be parallel Gel-based size selection to polish the dispersal of DNA piece and improve DNA library quality. At both ends of the DNA fragment, universal adapters are ligated which facilitates the PCR amplification with a single pair of primers or a collection of DNA fragments to stick to the surface by using adopter-complementary oligos. PCRs are run with rare rounds to avoid PCR biases and thus DNA fragments are prepared by individual technology which are called the fragment library. A couple of existing approaches are categorized on the basis of parameters to target enrichment. These include: specificity, consistency (sequence exposure across target sections), reproducibility, expense, convenience, and sample amount. The NGS has revolutionized the genome-wide studies and because of its multiplex nature and high throughput, it is capable of sequencing an entire human genome in a single day.

## 3.3. Datasets

A cluster of recurrently benchmark datasets[3] offer a helpful tool for testing new algorithms for FAP. Apart from FAP, sixteen other well-known benchmarks have been utilized in this work. These benchmarks are a standard for the comparative analysis in the current domain of study. These were produced from the sequences which can be retrieved from a freely accessible online repository, i.e., National Center for Biotechnology Information[4] (NCBI). Details of these datasets are mentioned in Table 2.

A set of 25 benchmark datasets is used for experiments in this work. Each instance of the data contain a number of fragments. These fragments are used to reconstruct the original sequence (from which these fragments are generated). A tool used in genomics termed as GenFrag 2.1 is employed to divide the first four sequences into an overlapping fragment. This tool requires a nucleotide sequence as an input which results in the synthesis of a set of overlapping fragments on the basis of built-in constraints. This is exclusively suitable for testing fragment assembly. However, GenFrag 2.1 may deliver somewhat diverse yield because of various random numbers being utilized. Therefore, it is not for sure that precise outcome will be obtained every single time. As a result of GenFrag, 10 GenFrag instances are generated, namely x601894, x601895, x601896, x601897, m154215, m154216, m154217, j024597, bx8425964, and bx8425967. Each instance has to cover four parameters such as mean fragment length, number of fragments, and the length of the original sequence. The sequences at point 5 refers to acin sequences and are comparatively lengthier in contrast to previously chosen sequences. These sequences are chopped down into fragments by a tool called DNAgen which were selected for their discrepancy in complexity levels.

As a result of DNAgen, six GenFrag instances are generated, namely acin1, acin2, acin3, acin5, acin7, and acin9 having four parameters, namely, coverage, mean fragment length, number of fragments, and the original sequence length. A group of 14 datasets known as the f-series are additionally included here. Adequate information is not available about these instances and their genetic makeup is vague. However, they are still utilized in the experiments as they can be used for benchmarking. The names of these f-series datasets are f25_305, f25_400, f25_500, f50_315, f50_412, f50_498, f100_307, f100_415, f508_354, f635_350, f737_355, f1343_354, and f1577_354. The available parameters of these instances are coverage and mean fragment length only.

[3] http://chac.sis.uia.mx/fragbench/index1.php

[4] https://www.ncbi.nlm.nih.gov/gds

PowerAwareLocalSearch ()
**Input:** Initial solution
**Output:** Modified solution *s* (in a systematic way)

```
s←generate initial solution
Repeat while there are no changes
   L←Ø
    for i=0 to N do
       for j=0 to N do
           $\Delta_c$, $\Delta_f$←CalculateDelta(s,i,j)
          if ( $\Delta_c$ $\geq$0 )
               L←LU<i,j,$\Delta_c$, $\Delta_f$>
           end if
        end for
    end for
    if L $\neq$ Ø
        <i,j,$\Delta_f$,$\Delta_c$>←select(L)
        Apply movement(s,i,j)
    end if
end repeat
return s
```

**Algorithm 1:** Power Aware Local Search

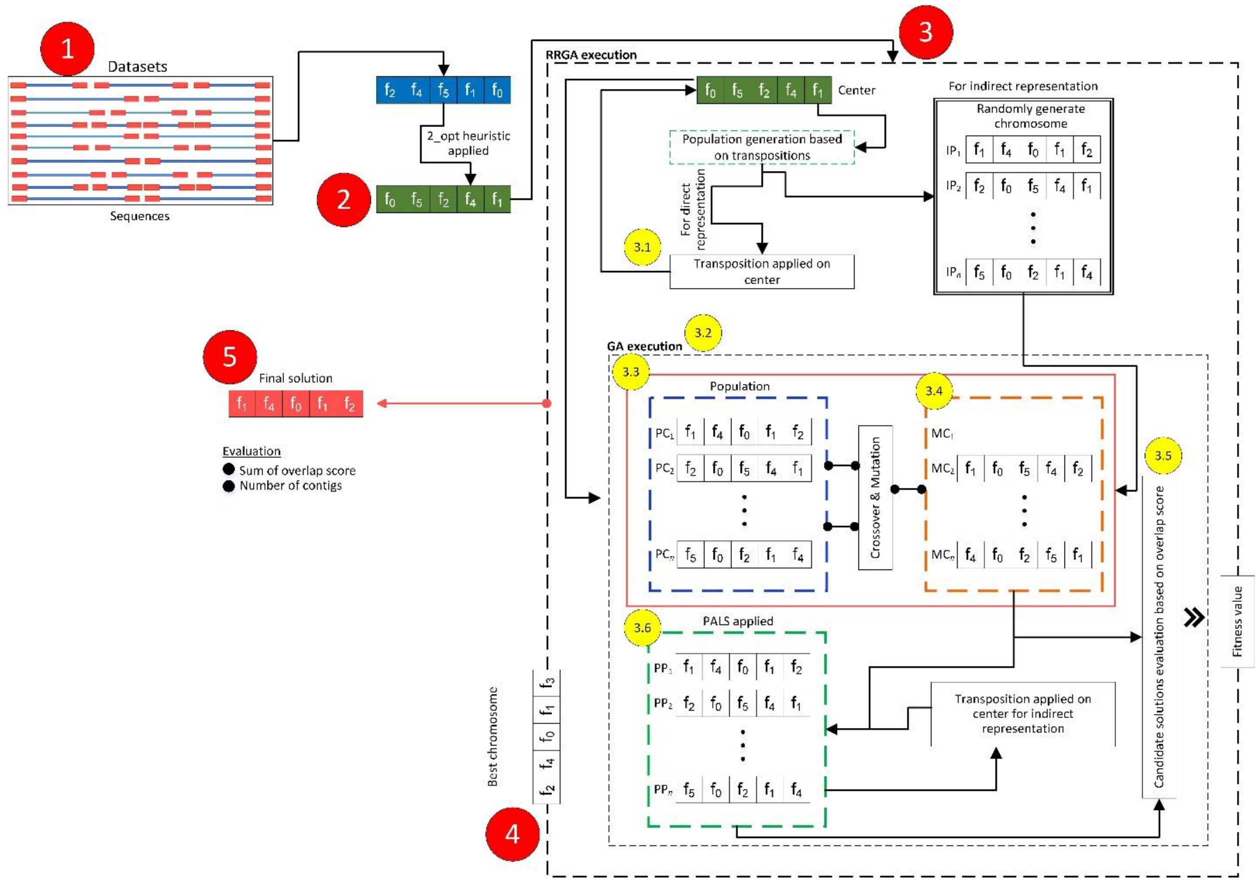
CalculateDeltaFunction()
**Input:** Solution (chromosome / individual) and two indexes *i* and j
**Output:** Δc Δf (variation of contig Δc and variation of overlap score Δf)

```
$\Delta_c$←0
$\Delta_f$←0
$\Delta_f$=$w_{s[i-1]s[j]}$ + $w_{s[i][j+1]}$
$\Delta_f$←$\Delta_f$ -$w_{s[i-1]s[i]}$ + $w_{s[j][j+1]}$
if $w_{s[i-1]s[i]}$ > cutoff
   $\Delta_c$=$\Delta_c$+1
end if
if $w_{s[j]s[j+1]}$ > cuttoff
   $\Delta_c$=$\Delta_c$+1
end if
if $w_{s[i-1]s[j]}$ > cuttoff
   $\Delta_c$=$\Delta_c$-1
end if
if $w_{s[i]s[j+1]}$ > cuttoff
   $\Delta_c$=$\Delta_c$-1
end if
return Δc Δf
```

**Algorithm 2:** Calculate Delta Function

## 4. Proposed solution

This section presents the proposed solution for the DFA. The proposed evolutionary computing-based framework comprises of six primary components, namely, PALS, sequence alignment, representation, transposition, RRGA, and 2-Opt. All of these components complement each other. The PALS is used to order the fragments while reducing the number of contigs. It is used as an evolutionary operator in the proposed framework. For sequence alignment, Smith-Waterman algorithm is utilized to compute the overlap score between the adjacent fragments. Transposition is used



1
Datasets
Sequences
2_opt heuristic applied
2
3
RRGA execution
Center
Population generation based on transpositions
For direct representation
3.1
Transposition applied on center
For indirect representation
Randomly generate chromosome
3.2
GA execution
3.3
Population
Crossover & Mutation
3.4
3.5
Candidate solutions evaluation based on overlap score
Fitness value
3.6
PALS applied
Transposition applied on center for indirect representation
Best chromosome
4
5
Final solution
Evaluation
Sum of overlap score
Number of contigs

**Fig. 2.** Complete working of the proposed evolutionary computing-based framework.

to generate candidate solutions for the evolutionary computing-based framework population. Where, these transpositions can produce both direct and indirect representation. The RRGA is utilized for the convergence purpose. To start the RRGA with a better initial population, 2-Opt heuristic is used. Fig. 2 demonstrate complete working of the proposed framework. Where, the red and yellow circles indicate sequence of various steps involved in the proposed evolutionary computing-based framework.

The proposed solution is based on the concept of metaheuristics. Metaheuristic is a problem independent master strategy that provides a guiding method for enhancing the performance of other heuristics. Finding the feasible solution for various optimization problems is intractable. Therefore, heuristics and metaheuristic procedures are used for designing better solutions. Significance of metaheuristic is based on two concepts; intensification and diversification. Intensification intends to search new solutions while focusing on the current best way to report the situation. Whereas, diversification proposes to create diverse solutions to explore the search space on a global scale. The balance between intensification and diversification is the characteristic of metaheuristics while selecting the best solutions. This section starts with a brief introduction of the six primary components and other related concepts utilized in the proposed work. Next, sequence alignment coupled by its types is discussed, followed by the core of the proposed solution with relevant explanation.

## 4.2. Individual components of the solution

Before explaining the proposed solution, it is important to explain the working of a few of the prerequisite techniques employed in this proposal. These include: PALS, sequence alignment methods, and Smith-Waterman algorithm.

**Power Aware Local Search:** The PALS metaheuristic [34] is a variation of the Lin's 2-opt method, that explores the neighborhood of a tentative solution. The key idea underlying PALS is that instead of evaluating the complete fitness values, the neighbors are exploited efficiently by the computation of a lighter fitness function. This is done based on

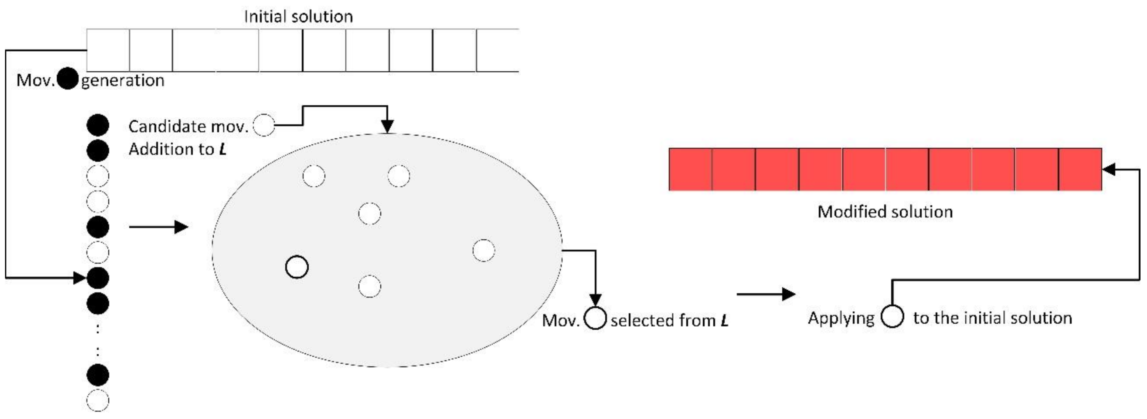


**Fig. 3.** The PALS process overview.

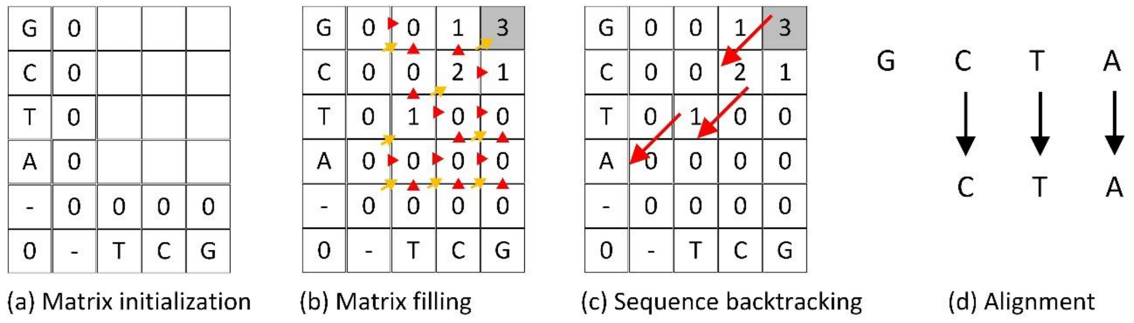


**Fig. 4.** Local sequence alignment.

the changes in the tentative solution. This method analyzes a large number of candidate solutions so that more comprehensive analysis of the search space is carried out and a high-quality result is obtained. In each iteration, the systematic explorative search over the whole neighbored of a tentative solution is efficiently carried out. Afterwards, only the candidate movements are retained. The candidate movement is used as a variation operator applied to the tentative solution. The mechanism continues till the end of new candidate movement production.

PALS iniatially works on a single solution generated by producing the initial solution (see Algorithm 1). Later, the solution is changed iteratively using movement. Movement is a perturbation (see Algorithm 2) that gives two indices $i$ and $j$ and a solution $s$. Theses indices reverses the subpermutation (the fragments order) between the two positions, i.e., $i$ and $j$ in the solution. For the DFA problem, while applying the movement, the principle step is to find the varation of contig (Δc) and fragment overlap (Δf) between the present solution and the solution obtained after applying the movement. The calculation is computationally inexpensive because instead of finding the fitness function or the number of contigs, it finds the varation of the contig and overlap. To estimate these values, this work analyze the affected fragments by the tentative movement ($i$, $j$, $i$-1, $j$+1). Adding the modified solution and removing the overlap score of the current solution, effect the fragments Δf (see algorithm 2). This analyzes whether some contigs are broken or merged by the movement operator.

For all possible movements in each iteration, PALS does all these calculations. A list $L$, stores all movements that reduce the number of contigs ($\Delta_c \leq 0$). The movement is selected from a list $L$ after completion of all possible calculations selecting the movement from a list having low number of contigs. In a situation when there are multiple movements having the same number of contigs, movement of higher Δf value is applied. The process continues until no candidate movement is further generated. Fig. 3 shows typical working of the PALS.

**Sequence alignment:** Sequence alignment is another term used for comparative sequence analysis that is one of the fundamental technique used in structural bioinformatics. The basic concept of sequence alignment is to align the DNA/RNA or protein sequences in a regular fashion to identify the conserved regions. The elementary goal of

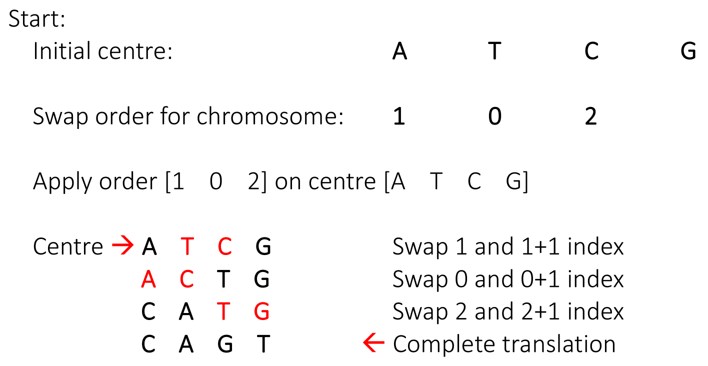


**Fig. 5.** Portrayal of translation algorithm required for the indirect and direct representation.

alignment is to deduce structural and functional relationships among the proposed sequences along with its evolutionary connections that may be termed as phylogenetic analysis. An algorithm based on dynamic programming approach is utilized that creates sub-problems from the main problem. These are independent and scores are assigned to each unit that depends on its similarity according to the parameters adjusted for a particular case. The structure and function of a novel sequence can be predicted via sequence alignment. This is because it is claimed that species under the same genes would have higher similarity among their genomic or protein sequences. Higher rate of the conservancy in sequence denotes greater similarity of both structure and function. There are two approaches used for sequence alignment, namely, Global Sequence Alignment (GSA) and Local Sequence Alignment (LSA). The GSA approach is usually employed for comparative investigation of homologous genes or sequences that are of almost equal dimensions and have some sort of relativity. Here, the alignment is performed since beginning till the last residue of the sequence (globally aligned) to identify the conserved regions/hotspots. Whereas, the sequences that are not for sure to have resemblance or even unrelated sequences can be aligned with LSA technique. LSA is utilized in discovering preserved region in DNA sequences, conserved domains or motifs among multiple protein sequences. Both these alignment approaches are typically defined by the dynamic programming method for aligning two dissimilar series of residues.

**Smith-Waterman algorithm:** It is a dynamic programming-based LSA algorithm [35] utilized to calculate the overlap metrics. Common parameter settings for this are, -2 for gap, for a match 1 is used and for mismatches, it uses -3. This algorithm finds the overlap scores in both directions for every combination of fragments. This is because the best possible orientation is unidentified yet. Once the algorithm calculates the overlap score between the fragments, it is placed in an overlap matrix. This algorithm contains the following three steps: (a) matrix initialization, (b) filling the matrix with the suitable scores, and (c) sequence backtracking.

For matrix initialization, consider the following two sequences for LSA.
ATCG (sequence #1, *X*)
  TCG (sequence #2, *Y*)
These are organized in a matrix form with *X*-2 rows and *Y*-2 columns. The values in the row *N*-1 and second column are set to zero as shown in Fig. 4. Where, *x*, *y* represent rows and columns. The matrix cell value is denoted by $M_{xy}$, *C* is the score of the required cell $C_{x,y.}$ In case of match or a mismatch, the gap alignment is symbolized by *G*.

$$M_{x,y} = Max[M_{x+1,y-1} + C_{x,y,} M_{x,y-1} + G, M_{x+1,y} + G,\ 0] \tag{1}$$

RecenteringRestartingGeneticAlgorithm()
**Input:** Initial solution (Centre / Initial order of fragments)
**Output:** Solution *s* (correct order of fragments)

```
1. Assign initial centre $\sigma$
2. Set Flag
3. while Stopping Criteria is Not Met
4.    if FlagSet
5.        Center=$\sigma$
6.        Initialize Population Through use of Transpositions
7.    end if
8.    Execute Genetic Algorithm
9.    if solution $\tau$ is better than $\sigma$ then
10.        $\sigma$=$\tau$
11.        Set Flag
12.    else
13.        Increase/Decrease number of transpositions to be generated
14.        Reset Flag
15.    end if
16. end while
```

**Algorithm 3:** Recentering-Restarting Genetic Algorithm

Once the matrix cell value is filled, the next step is filling up the entire matrix. It is important to have the knowledge of neighboring values (diagonal, upper, and left) of the current cell to fill up the entire matrix. After initiating the matrix, the first character (nucleotides) of both sequences are compared, such as 'T' and 'A', in case of a match or mismatch the score is added with the diagonal value ($M_{x+1,y-1}$) of $M_{x,y}$. However, the gap value is added to the left ($M_{x,y-1}$) and down ($M_{x+1,y}$) values of the cell ($M_{x,y}$). Therefore, the mismatch value -3 is added to the diagonal and a gap penalty of -2 is added to the left and down value. The value for $M_{x,y}$ is calculated using Eq. (1). The LSA matrix only contain bits, therefore the negative values are replaced by 0 and positive with 1. The entire cells are filled up using this approach. A sample filled matrix can be seen in Fig. 4. Where, each cell is pointed by one or more neighboring cells, indicating the maximum score obtained from a particular cell.

Before applying the final step, back tracing is needed to find the maximum value in the entire matrix. The maximum value can be in multiple cells. In that case, there is a possibility of multiple alignments. The best alignment is selected based on the scoring. In the example (Fig. 4), the maximum score in the matrix is 3. Consequently, the trace begins from the locality that is having the highest score which points back using the pointers, finds the possible predecessor and moves on to the next one. This process continues until a score of 0 is obtained (Fig. 4(c)). Thus, a possible local alignment is obtained as shown in Fig. 4(d).

**Representation:** Two types of representations based on transposition for the RRGA are available. These include a direct and the indirect representation. For the direct representation, the centre is selected either randomly, or with some seed. To generate the population of *m* orderings (chromosomes), this work generates *m* lists of numbers of length *n*. Swaps (transpositions) are applied to a copy of the centre and once all *n* swaps have occurred, the chromosome is finally created. This is done for all *m* chromosomes. For the indirect representation, like the direct representation, a centre is selected and the population is generated. The only difference for indirect representation is that the population does not contain the orderings, but comprises of the list of swaps. Swaps application step is skipped when creating the population. A population of size *m* is still created, however, the chromosomes are the lists of swaps of length *n*. These chromosomes with the alphabet of positive integers less than the length of the centre are then evolved. There is a catch, however, with this representation. In order to evaluate the fitness of the chromosomes, it is needed to actually translate the chromosomes of swaps to an ordering like the one mentioned in the direct representation. The swaps are actually applied to a copy of the centre at the time of fitness evaluation. This transposition is created as follows [16]. Suppose, $a_1, a_{2,} \dots \dots a_{n,}$ is an ordered list of numbers $\{1,2,3 \dots n\}$. The set of transpositions that exchange the pairs $a_i$, $a_i$+1 generate $S_n$ (they form a generating set). This generating set is of size *n*-1.

***Transposition:*** It is a step that precisely trades two items while leaving all others unaltered. Consider a centre of four letters for simplicity: A, T, C, and G. To generate 5 chromosomes, 5 lists of swaps will be created having arbitrary length, let's say 3 in this case. The numbers in the list of swaps have to be between 0 and the length of the center. Fig. 5 shows the translation required for the indirect and direct representation.

Orderings:
0, 1, 2
0, 1, 1
1, 0, 2
2, 2, 2
2, 1, 2

These swaps are then applied to the copies of the centre.
T, C, G, A
T, A, C, G
C, A, G, T
A, T, G, C
A, G, C, T

## 4.3 Recentering-Restarting Genetic Algorithm

Primary feature of the RRGA is its ability to escape the local optima. This is the reason RRGA is employed in various past works [36], e.g., TSP, studying epidemic networks, and side effect machines for decoding. RRGA has this ability due to its sequential search through the entire search space. This produces global optimum while avoiding the local optima. The algorithm starts by selecting the starting center. The centre is the initial representation of a solution to the problem. For a given problem, this representation might be chosen randomly or by utilizing some heuristic approach. Once the centre is chosen, the population construction would be convenient. The population is created based on two types of representation from the center: a direct representation and the indirect transposition representation. Algorithm 3 lists the working of RRGA.

After generating the population, the next step is to execute a model GA. The GA runs for a specified number of times and after its completion the obtained best fitness value from the GA is compared with the fitness of the current center. If the fitness value of the best chromosomes is greater than the current center, it will be considered as new centre Subsequently, the entire procedure is repeated. This may cause the number of transpositions to either increase or decrease. The expanded quantity of transpositions increases while the current centre remains unchanged in case if the fitness value of the best chromosome is not enhanced and vice versa. The search space is different for the next iteration when the centre is changed while maintaining the gain from the previous run. The two variants of the algorithm exploit the producing set by making their phenotypes from its transpositions. The utilization step is in fact the difference between the two variants in this algorithm. The direct representation uses the transposition right before the evaluation step, while indirect transposition representation practices the transposition during the chromosome evaluation step. The stopping criteria of GA usually is to recognize that the learning has stopped because of convergence [37]. The search space of the fitness landscape for an evolutionary algorithm is same but distinct from the centre because of the transpositions' nature**.** After recentering, if the algorithm finds local optima, the search continues with other regions of the fitness landscape [38].

2-optSwap()
**Input:** initial permutation of fragments, *k*, *m* (*k* is index of a selected fragment and *m* is length of permutation/solution)
**Output:** final permutation of fragments

```
1. Select permutation[0] to permutation[k-1] and add in order to new_permutation
2. Select permutation[k] to permutation[m] and add in reverse order to new_permutation
3. Select permutation[m+1] to end and add in order to new_permutation
4. return new_permutation
```

**Algorithm 4:** Swap procedure of 2-opt

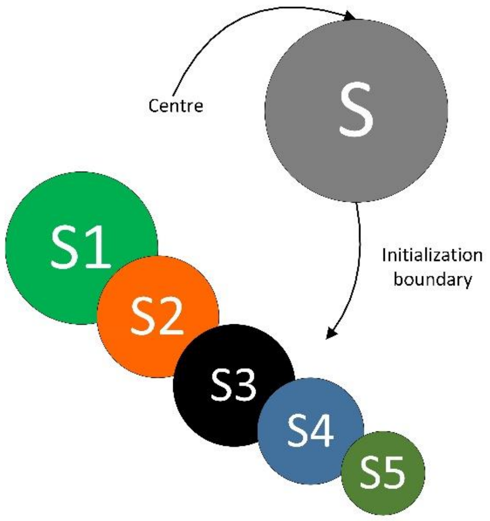


**Fig. 6.** Sequential search space traversal demonstrated by RRGA.

After each execution of the traditional GA, the number of transpositions are altered that are produced by generating sets. The algorithm reboots in case no learning is detected after a few cycles. Thus, the number of reboots for a particular run can be dynamically defined. The primary amount of transpositions can be defined as some division of the solution length. In RRGA, after the GA is run every single time, there are two different possible ways for the number of transpositions to be refashioned. The number of transpositions is amplified by a fixed predefined numeral in case of no improvement or it is decreased by a fixed predefined number. The number of transpositions experience quick fluctuations because of the dynamic behavior of the algorithm. To cope with this situation, upper and lower limits are defined for the number of transpositions. If this limit is crossed, the number of transpositions is reset to the initial value. Although any value can be used, reasonable limits are 10 times and 1/10th of the initial value. Sometimes it may be advantageous to force a recentre after the first GA run. This promote distance between the initial seed and the centre to help remove the risk of stagnation on the seed's local optimum. Fig. 6 depicts the sequential traversal through the search space of the RRGA.

**4.4. 2-Opt**

The heuristic called 2-opt is specifically designed for TSP [39, 40]. It is also casted on for other various similar applications. The success of 2-opt has encouraged the employment of 3-opt, *k*-opt, and some other algorithms as well as the dynamic approach of Lin-Kernighan. The main step in 2-opt involves the selection of a sub-permutation and reversing it within the permutation. The newly generated permutation is compared to the previous best permutation and finally the best permutation is achieved. The approach listed in Algorithm 4 is used by 2-opt to compute permutation.

**4.5. Contig**

Contig is a layout consisting of contiguous overlapping fragments. It is a sequence in which the overlap between adjacent fragments is greater than a predefined threshold. Calculating a number of contigs ensure the order of the fragments in term of cutoff/threshold. The *CalculateDelta* function is used to set the overlap between two adjacent fragments. Classical assemblers practice fitness functions that favor solutions in which strong overlap occurs between adjacent fragments of the layouts. The formula for overlap score is given in Eq. (2).

$$F_{score} = \sum_{i=0}^{n-2} w_{f[i],f[i+1]} \tag{2}$$

Where, $w_f$ represent the overlap score between two adjacent fragments $i$ and $i$+1.

A cutoff value is set in the *CalculateDelta* function of Algorithm 2. The cutoff, that is fixed up to 30 (a very high value), provides one filter for spurious overlaps introduced by experimental error.

**4.6. Coverage**

The number of bases that individually exist in all of the pieces over the whole stretch of the DNA sequence is termed as coverage. On average, one can have an expectancy of ten instances for a provided base from an unknown spot from the template contig within a group of fragments for that contig. To ensure that multiple pieces of DNA contain the

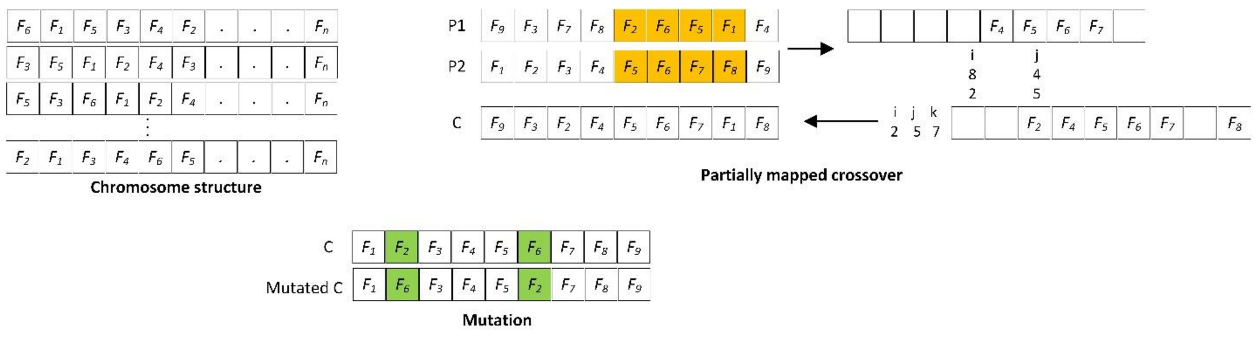


**Fig. 7.** Chromosome representation, crossover, and mutation procedures.

same base, the average coverage must be greater than 1 in case of FAP. Existence of single base in multiple fragments enhance the chance of fragments overlap which are vital for fragments order to be made possible.

$$\text{coverage} = \frac{\sum_{i=1}^{n} frag\ i\ len}{original\ sequence\ len} \tag{3}$$

Equation (3) represents that the coverage is the ratio of the sum of all the fragment's length and the original sequence's length.

## 4.7. The proposed solution core

A number of strategies have been used for the DNA fragments assembly problem, such as greedy graph-based algorithms, de Bruijn graphs, and the OLC approach. Focus of the current proposal is on OLC approach for this problem. The challenging step in OLC is the layout phase. In the layout stage, the permutation use each fragment only once to find correct order of the fragments. The idea proposed here is to define a hybrid approach where RRGA uses PALS as an evolutionary operator. One of the RRGA's primary feature is its capability of avoiding the local optima. It is because of the RRGA's traversal over the exploration space, along with the nature of the compulsory dynamic illustration. This work uses the classical measure such as the overlap score between adjacent fragments for the judgment of the assembly quality. However, the actual objective is to obtain an order of the fragments that minimizes the number of contigs. This has the goal of reaching one single contig, i.e., a complete DNA sequence that comprises of all the overlapping fragments. In PALS, the number of contigs are used as a high-level criterion to judge the result quality. However, the calculation of number of contigs is quite time-consuming, and this fact precludes any algorithm to use such calculations. A solution to this problem is to introduce methods that do not know the exact number of contigs and are computationally inexpensive. The key contribution of PALS is to indirectly estimate the number of contigs by measuring the actual number of contigs that are created or destroyed when tentative solutions are manipulated. To judge the actual quality of the DNA fragment assembly, this work uses PALS by calculating the number of contigs with RRGA to enhance the quality of DNA fragment assembly. The RRGA is executed with an initial representation of the fragments order, called centre. Centre represents the default order of the fragments in the dataset. The centre is then optimized by using 2-opt heuristic. Once the centre is generated, a set of chromosomes is created using the representation mechanisms. Afterwards, the GA+PALS are executed on the sets of chromosomes. Next, the reproduction operators, i.e., partially mapped crossover and swap mutation are applied to the chromosomes. The set of best chromosomes is obtained from GA+PALS. The best chromosome (B-chromosome) is selected based on fitness value from the set of chromosomes. Comparison of the B-chromosome is performed with the centre based on the fitness value. In case fitness value of B-chromosome is greater than the center, the number of transpositions are decreased by 5% and the centre is replaced with B-chromosome. This process is then repeated (generating the chromosomes, running the GA+PALS, and comparing the best chromosome with the center). However, if B-chromosome is not better than the center, number of transpositions are increased by 10% and the process is repeated without changing the current center. Due to high computational time of RRGA, the process is executed for 100 iterations. Stopping criteria and number of iterations for the other four competing methods during the experiments are also kept the same for a fair comparison. The fitness value used in the proposed solution is the overlap score. It must be noted that while computing the overlap score (i.e., the fitness value), number of contigs are also computed internally. The individuals with maximum overlap scores survive in the next generation. For the proposed idea, the evaluation is done on 25 benchmarks datasets. Comparison is based on the evaluation metrics of overlap scores and number of contigs. A solution is considered best on the basis of maximum number of overlap scores and the minimum number of contigs. A comparison is done with four state-of-the-art methods for the same task, namely, RRGA variation for DNA fragment assembly [16], PALS [27], GA [21], and hybrid GA [26].

**Fitness Function:** The proposed methodology assembles the reads based on two objectives, i.e., (a) reducing the number of contigs and (b) maximizing the sum of overlap scores. The fitness function utilized in this work is based on these objectives. The purpose of reducing the number of contigs is to order the reads in such a way that it becomes a single contig for which a threshold is set between the overlap scores. PALS is used as an evolutionary operator to minimize the number of contigs. Hence, reads in the individuals are ordered based on the first objective. The sum of overlap scores is used as a fitness value for the second objective. The individuals having better fitness value are passed on to the next generation. The surviving individuals are the ones having reads ordered by minimizing the number of contigs and having the maximum sum of overlap scores. Computation of the fitness function used in this work is listed in Eq. (2).

Suppose a set of fragments is an array $F$, where any fragment is represented by $F[I]$. The symbol $F[I]$ represents the $i^{th}$ fragments. The overlap score between any two adjacent fragments $f_{rm[i],}$ $f_{rm[i+1]}$ represent the similarity between the two fragments which is estimated by Smith-Waterman algorithm. Where, the fitness function is the sum of the overlap score between all adjacent fragments. The criteria of overlap and number of contigs is based on assessing the fitness value by considering the sum of the overlap score of each individual and also reflecting the number of contigs in an individual with the help of PALS. The number of contigs is also taken into consideration for the final sequence obtained. The significance of contig calculation is to ensure that the solution best represents a sequence that is continuously assembled. The fitness function in this work utilizes both the number of contigs and overlap score.

**Chromosome structure:** The DNA fragment assembly is a permutation problem. The chromosome is generated by either direct or indirect transposition representation technique. The chromosome is the list of fragments. The cells in the chromosome represents the fragment. The length of the chromosome represents the number of fragments in the FASTA dataset which is the same for each chromosome. Structure of the chromosome used in this work is shown in Fig. 7.

**Crossover and Mutation:** The partially mapped crossover method is used in this work. The process starts by randomly selecting any two chromosomes (as parents). Next, the partially mapped crossover algorithm is applied on two chromosomes to generate the resultant child. Once the two parents are selected, a substring from each parents is randomly chosen and is exchanged between the two. The mapping relationship is determined between the two subgroups. Later on, the relation map is utilized to legalize the offspring. The detail of the partially mapped crossover according to the current problem is explained as follows. For partially mapped crossover, an offspring is built by choosing a subsequence of elements from one parent. This preserves the order and positions of as many elements as possible from the second parent. The subsequence of elements that chunk its copy from the first parent to the child is selected by choosing two random cut points, which serves as a boundary for the swapping operations. The partially mapped crossover algorithm for the current problem has following steps.

- Start by randomly selecting any two chromosomes (as parents). Say, P1 and P2 are two chromosomes and C is the child (see. Fig. 7).
- Choose random segment and copy it from P1 to C.
- Then, start with the first crossover point and look for elements in that segment of P2, so in the corresponding segment of P2, it finds the element which has not been copied yet.
- For each of these elements called $i$ elements, look in the offspring to see which element $j$ has been copied in its place from P1, now its dealing with elements and its places. For each element $i$ , these are the elements in the segment of P2. It looks in the offspring and see which elements have been copied in its placed from P1, calling those elements the $j$ elements.
- Place $i$ into position occupied by $j$ in P2, as $j$ is already in the offspring.
- If the place occupied by $j$ in P2 has already filled in the offspring by an element $k$, it will put $i$ in the positions occupied by $k$ in P2. The $i$ correspond to j, and $j$ correspond to $i + 1$ and $i + 1$ goes to $k$. If $i$ and $k$ are equal elements, then $i$ will be compared with the next corresponding value and this process will continue until different values come across.
- The rest of the offspring is filled directly from P2 (second parent).

DNAFragmentAssemblyviaRecentringRestaringHybridGeneticAlgorithm()
**Input:** Initial order of the fragments (generate initial solution)
**Output:** Overlap score and the number of contigs of final solution

```
1. Representation: First, represent the initial order of fragments according to
their original order of sequence in the benchmark.
2. Initial centre of DNA fragment assembly: Apply the TSP heuristic called 2-Opt
on initial order $\varepsilon$ for optimizations.
3. While termination condition is not meet execute RRHGA
      Generate the population of size N, randomly select N order and then apply
      N transposition to the centre so as to generate the set of chromosomes.
4.    Execute GA with PALS (Until it does not meet the stopping criteria)
5.           Selection for crossover
            Apply tournament selection to a population in order to select the
            chromosome for crossover, and then apply partially mapped crossover
            on selected chromosomes.
6.           Mutation operator
            Apply swap mutation to chromosomes.
7.           Use PALS as evolutionary operator
            Use PALS as evolutionary operator to optimize the solution.
8.    Getting the best solution
      The best solution is obtained via Hybrid GA.
9.    Compare the solution with initial centre
      If a solution $\alpha$ after applying  GA is better than starting centre $\varepsilon$ .Then
      replace the initial centre with best solution.
               $\varepsilon = \alpha$
10.   Fluctuating the number of transpositions
      In case of improving the solution, decrease in the transposition by 5
      present is idle, otherwise an increase of 10 percent is recommended.
11.   Repeat step 4 to 10
      The process is repeated from step 4 to 10 continuously until the termination
      criteria is not satisfied.
12.   Overlap score and the number of contigs as a final output.
```

**Algorithm 5:** Pseudocode of DNA fragment assembly via recentring-restaring hybrid genetic algorithm

For the mutation operation (see Fig. 7) the swap mutation option is utilized in this work. Given an input chromosome, two elements are picked randomly to be swapped. This method preserves most of the adjacency information. Therefore, no major change happens in the chromosomes. Fig. 7 shows the crossover and mutation operations utilized here. Algorithm 5 lists the complete working of the proposed solution.

## 5. Experiments and results

This section lists the experiments performed to evaluate the current proposal and their results. For this, 25 benchmark datasets, including 14 f-series sequences, are utilized. The results are compared with four state-of-the-art approaches from the same domain, namely, RRGA variation for DNA fragment assembly [16], PALS [27], GA [21], and hybrid GA [18]. Three different types of experiments are carried out. These set of experiments not only assist in evaluating the current proposal, but also help in determining the optimum settings of the proposed evolutionary computing-based framework to solve the DFA problem. The three set of experiments are: (a) setting the initial fragment positions optimized through 2-opt, (b) utilizing PALS for optimizing the solution obtained after executing GA, and (c) setup with swapping GA representation. Each of the three set of experiment is performed with and without force recentre. Table 3 lists the parameter settings for the proposed work. Partially mapped crossover and swap mutation is used here as genetic operators. The selection of chromosomes is done by the tournament selection method. All the evolutionary algorithms are basically probabilistic methods and the output achieved in one iteration may not be obtained in the other. Therefore, while reporting results of such methods average value of multiple runs is usually reported. The same procedure is adopted here and an average of 10 runs is used.

**Table 3**
Parameters for conducting the experiments for DNA fragment assembly.

| | |
|---|---|
| Population size | 100 |
| Mutation rate | 2% |
| In case of no improvement, the number of transpositions is increased by | 10% |
| In case of improvement, the number of transpositions is decreased by | 5% |
| Cutoff value | 30 |
| Stopping criteria | No. of iterations, i.e., 100 |
| Objective function | Sumo of overlap scores |
| GA operators | Partially mapped crossover & swap mutations |

## 5.1. Experimental design

DNA fragment assembly is the process of reconstructing the original DNA sequence from small fragments of the DNA. Focus of the proposed work is on the OLC. The challenging part of OLC is the layout phase. OLC is a permutation problem in which each fragment is used exactly in an order of the layout that best represents the original sequence. The proposed work initializes the fragment positions utilizing 2-opt to enhance the solution [31]. The goal of the proposed solution is to maximize the overlap score of the layout and decrease the number of contigs to generate a high-quality solution.

The overlap score of a solution is the sum of overlap scores between all the adjacent fragments *i* and *i*+1 in the permutation that can be calculated via Eq. (2). A better solution is the one with a higher overlap score. The second objective is to decrease the number of contigs with the help of *CalculateDelta* function (PALS algorithm). The contig finds the order of the overlapping fragments based on a threshold. The PALS algorithm optimizes the permutation by adjusting the order of the fragments for creating or destroying the number of contigs by generating the tentative solution [41]. Once the solution is optimized, candidate solutions are evaluated for their fitness value. The proposed solution analyzes the final sequence by calculating the number of contigs along with their fitness value. Various benchmarks exist for the comparison of existing methods and new approaches. Generally, eleven of these benchmarks are frequently used, namely, x601894, x601895, x601896, x601897, m154215, m154216, m154217, j024597, bx8425964, bx8425967, and acin1. Comparison of the proposed solution with the four state-of-the-art methods is carried out using these benchmarks and fourteen f-series datasets (see Table 2).

Three types of experiments are conducted here. The first set of experiments consist of the initial fragment position which is optimized by 2-opt [16]. Due to the dynamic nature of RRGA, the algorithm restarts many times after a few generations in case of no learning. The effect of recentreing after the first restart is also studied. The effect of using PALS as a genetic operator for optimizing the individuals is analyzed. The second set of experiments repeat the first one, however, instead of using PALS as a genetic operator, it is utilized for optimizing the solution obtained after executing the GA. The third set is the replica of the first one with the difference that the representation for the GA is swapped if the first run of GA uses direct representation. The second one then uses the indirect representation and vice versa [16]. A comparative analysis is carried out with the four best algorithms shown in Table 4. The three types of experiments are validated by force recentering. This is because sometimes it maybe beneficial to force a recentre right after the first GA execution, which indorses the path length from the starting path to the center. Therefore, enabling to avoid the hazards of inertia over the seed's local optimum.

## 5.2. Experiments with initial fragment position optimized through 2-opt

All tables shown in the experiments represent the results of both direct and indirect transposition representations. The tables show the results of 11 problem instance with 14 f-series datasets. Therefore, there are 25 datasets in each table. Each dataset is evaluated based on their obtained fitness value and number of contigs. The first set of experiments is the same as in [16], i.e., dynamic restart, force recentering after the first restart, altering the percentage of *n* transpositions. The improvement in this work is achieved by utilizing the PALS as a genetic operator. Two objective functions are used to evaluate the final sequence; the overlap scores and number of contigs. The contig values presented in the results here are calculated via PALS algorithm. The negative contig value denotes that the number of adjacent fragments is higher than a predefined threshold which is decreased by -1 as shown in Algorithm 2 (step 15). This process keeps on decreasing the contig's value and finally it becomes less than zero where it is represented in the form of negative integers.

**Table 4**
Summary of results on all 25 datasets using comparison papers.

| | RRGA | | | GA | | | GA+PALS | | | PALS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Fitness value | Contig | Time | Fitness value | Contig | Time | Fitness value | Contig | Time | Fitness value | Contig | Time |
| m15421_5 | **22143** | **-20** | 2.00E+09 | 7450 | 90 | 2.00E+09 | 6181 | 96 | 2.00E+09 | 80 | 259 | 2.00E+09 |
| m15421_6 | 28704 | -42 | 2.00E+09 | 7501 | 132 | 2.00E+09 | 5236 | 148 | 2.00E+09 | 79 | 289 | 2.00E+09 |
| m15421_7 | **34463** | **-44** | 2.00E+09 | 6455 | 146 | 2.00E+09 | 5868 | 150 | 2.00E+09 | 80 | 259 | 2.00E+09 |
| x60189_4 | 5726 | 0 | 2.00E+09 | **6793** | **2** | 2.00E+09 | 3842 | 16 | 2.00E+09 | 91 | 306 | 2.00E+09 |
| x60189_5 | 8731 | -11 | 2.00E+09 | 2917 | 25 | 2.00E+09 | 3798 | 23 | 2.00E+09 | 81 | 259 | 2.00E+09 |
| x60189_6 | **10643** | **-11** | 2.00E+09 | 3747 | 45 | 2.00E+09 | 3659 | 35 | 2.00E+09 | 84 | 267 | 2.00E+09 |
| x60189_7 | 11670 | -5 | 2.00E+09 | 4805 | 23 | 2.00E+09 | 4910 | 23 | 2.00E+09 | 81 | 259 | 2.00E+09 |
| j02459_7 | **70701** | **-71** | 2.00E+09 | 16547 | 281 | 2.00E+09 | 14740 | 287 | 2.00E+09 | 70 | 259 | 2.00E+09 |
| acin1 | 46934 | -298 | 2.00E+09 | 43285 | -292 | 2.00E+09 | 3249 | -298 | 2.00E+09 | 71 | 253 | 2.00E+09 |
| bx842596 4 | 125070 | -59 | 2.00E+09 | 10848 | 409 | 2.00E+09 | 10718 | 411 | 2.00E+09 | 123 | 454 | 2.00E+09 |
| bx842596 7 | **246070** | **-152** | 2.00E+09 | 16983 | 730 | 2.00E+09 | 43580 | -298 | 2.00E+09 | 123 | 454 | 2.00E+09 |
| f25_305 | 1974 | 10 | 2.00E+09 | 1335 | 12 | 2.00E+09 | 1789 | 14 | 2.00E+09 | 124 | 502 | 2.00E+09 |
| f25_400 | 3530 | 4 | 2.00E+09 | 3249 | 6 | 2.00E+09 | 1770 | 12 | 2.00E+09 | 77 | 302 | 2.00E+09 |
| f25_500 | 5226 | -4 | 2.00E+09 | 3917 | 2 | 2.00E+09 | 3945 | 4 | 2.00E+09 | 124 | 502 | 2.00E+09 |
| f50_315 | **4018** | **17** | 2.00E+09 | 3053 | 23 | 2.00E+09 | 1948 | 29 | 2.00E+09 | 80 | 277 | 2.00E+09 |
| f50_412 | 6178 | 11 | 2.00E+09 | 4311 | 19 | 2.00E+09 | 4665 | 13 | 2.00E+09 | 101 | 384 | 2.00E+09 |
| f50_498 | **8583** | **1** | 2.00E+09 | 11331 | -1 | 2.00E+09 | 3481 | 13 | 2.00E+09 | 117 | 393 | 2.00E+09 |
| f100_307 | **7104** | **47** | 2.00E+09 | 6264 | 65 | 2.00E+09 | 2604 | 85 | 2.00E+09 | 85 | 295 | 2.00E+09 |
| f100_415 | 8697 | 53 | 2.00E+09 | 3802 | 89 | 2.00E+09 | 6576 | 67 | 2.00E+09 | 102 | 330 | 2.00E+09 |
| f100_512 | 11586 | 45 | 2.00E+09 | 3846 | 81 | 2.00E+09 | 4484 | 73 | 2.00E+09 | 140 | 466 | 2.00E+09 |
| f508_354 | 55046 | 322 | 2.00E+09 | 9562 | 491 | 2.00E+09 | 12413 | 471 | 2.00E+09 | 67 | 281 | 2.00E+09 |
| f635_350 | **57309** | **426** | 2.00E+09 | 15126 | 604 | 2.00E+09 | 15579 | 602 | 2.00E+09 | 97 | 336 | 1.53E+09 |
| f737_355 | 112546 | 726 | 2.00E+09 | 736 | 736 | 2.00E+09 | 14364 | 716 | 2.00E+09 | 101 | 739 | 1.53E+09 |
| f1343_354 | 123130 | 906 | 2.00E+09 | 21362 | 1330 | 2.00E+09 | 21657 | 1334 | 2.00E+09 | 68 | 294 | 2.00E+09 |
| f1577_354 | **123130** | **906** | 2.00E+09 | 24829 | 1566 | 2.00E+09 | 24828 | 1568 | 2.00E+09 | 91 | 341 | 2.00E+09 |

**Table 5**
First set of results with force recentre.

| Dataset | Direct |  |  | Indirect |  |  |
|---|---|---|---|---|---|---|
|  | Fitness value | Contig | Time | Fitness value | Contig | Time |
| m15421_5 | **21553** | **-8** | 1524926083 | 20476 | 0 | 1524963941 |
| m15421_6 | **30306** | **-50** | 1524908557 | 27931 | -32 | 1524992468 |
| m15421_7 | **33515** | **-39** | 1525008258 | 31065 | -18 | 1525009295 |
| x60189_4 | 6009 | -2 | 1524964126 | **7032** | **-14** | 1524984845 |
| x60189_5 | 8231 | -9 | 1524966285 | **8500** | **-9** | 1524989358 |
| x60189_6 | **9749** | **-5** | 1524966289 | 9136 | 5 | 1524940843 |
| x60189_7 | **12838** | **-15** | 1524963528 | 11894 | -7 | 1524955261 |
| j02459_7 | **68741** | **-59** | 1525008627 | 65708 | -37 | 1525090946 |
| acin1 | **46947** | **-298** | 1524448446 | 46651 | -286 | 1524991598 |
| bx842596 4 | **126113** | **-69** | 1525578659 | 122903 | -47 | 1525572736 |
| bx842596 7 | **245634** | **-160** | 1525618669 | 240446 | -130 | 1527432020 |
| f25_305 | 2161 | 4 | 1524922174 | **2288** | **4** | 1524920276 |
| f25_400 | **4132** | **-4** | 1524823823 | 3799 | 2 | 1524922122 |
| f25_500 | 5112 | 2 | 1524901379 | **5440** | **0** | 1524986792 |
| f50_315 | 3717 | 15 | 1525014442 | **4152** | **11** | 1525012912 |
| f50_412 | **6548** | **7** | 1525077368 | 6359 | 9 | 1525102491 |
| f50_498 | 7556 | 3 | 1524874474 | **8458** | **1** | 1524875744 |
| f100_307 | 5813 | 59 | 1532460882 | **6278** | **53** | 1532463052 |
| f100_415 | **8348** | **57** | 1524892166 | 8152 | 53 | 1524880752 |
| f100_512 | 11050 | 47 | 1524466104 | **11556** | **45** | 1525009131 |
| f508_354 | **44218** | **263** | 1524970014 | 43750 | 265 | 1525051146 |
| f635_350 | **55202** | **322** | 1524983434 | 54912 | 324 | 1524983630 |
| f737_355 | **56940** | **428** | 1524947267 | 55330 | 442 | 1525071886 |
| f1343_354 | **112246** | **728** | 1525127233 | 112009 | 730 | 1525084466 |
| f1577_354 | **123522** | **904** | 1525203170 | 121705 | 916 | 1525130401 |

**Table 6**
First set of results without force recentre.

| | Direct | | | Indirect | | |
|---|---|---|---|---|---|---|
| Dataset | Fitness value | Contig | Time | Fitness value | Contig | Time |
| m15421_5 | **22598** | **-14** | 1525648652 | 54932 | -2 | 1525556089 |
| m15421_6 | **29469** | **-46** | 1525421686 | 27774 | -30 | 1526817381 |
| m15421_7 | **32744** | **-38** | 1525108769 | 54932 | -28 | 1525856065 |
| x60189_4 | 6488 | -8 | 1525395759 | **6324** | **-4** | 1525549836 |
| x60189_5 | 8655 | -11 | 1525822448 | **8496** | **-5** | 1525397582 |
| x60189_6 | **9943** | **-7** | 1525290915 | 9652 | -1 | 1525328120 |
| x60189_7 | **11546** | **-3** | 1525332860 | 11160 | 3 | 1525333085 |
| j02459_7 | **68736** | **-59** | 1525566386 | 66109 | -37 | 1525461427 |
| acin1 | **47037** | **-298** | 1525334606 | 47043 | -298 | 1525334596 |
| bx842596 4 | **125711** | **-65** | 1526844963 | 122943 | -47 | 1526838077 |
| bx842596 7 | **247856** | **-160** | 1533132218 | 241509 | -130 | 1530581210 |
| f25_305 | 2271 | 4 | 1525348557 | **2124** | **8** | 1525363178 |
| f25_400 | **3139** | **4** | 1525423209 | 4024 | 0 | 1525363943 |
| f25_500 | 5777 | -4 | 1525440028 | **5143** | **0** | 1525467617 |
| f50_315 | 4013 | 19 | 1525967670 | **4485** | **13** | 1525005277 |
| f50_412 | **5835** | **17** | 1526257288 | 6355 | 3 | 1526077729 |
| f50_498 | 9050 | -3 | 1525873508 | **8100** | **-1** | 1525783248 |
| f100_307 | 7035 | 43 | 1532469479 | **7111** | **53** | 1532468245 |
| f100_415 | **9202** | **51** | 1525833169 | 8461 | 53 | 1525833177 |
| f100_512 | 11881 | 41 | 1525577068 | **54932** | **47** | 1525576977 |
| f508_354 | **44240** | **263** | 1525584043 | 43589 | 267 | 1525583997 |
| f635_350 | **55137** | **324** | 1525487810 | 54932 | 324 | 1525490303 |
| f737_355 | **57525** | **426** | 1525661171 | 55366 | 442 | 1526309255 |
| f1343_354 | **113172** | **722** | 1525649599 | 111753 | 732 | 1526854255 |
| f1577_354 | **123446** | **916** | 1526818991 | 121733 | 916 | 1525684222 |

### 5.2.1. Forced recentre

Table 5 shows the results on 25 problem instances for force recentre approach with both representations. The results show best fitness value achieved by the present work for 5 out of 11 problem instances in comparison to the competing algorithms. While for 3 out of 14 f-series instances, the results are better using the proposed solution. For comparing the contigs, the result shows minimum and same number of contigs for 4 and 3 f-series instances, respectively. The results for problem instance with minimum and same number of contig are 2 and 2, respectively. Therefore, the overall summary of the results for the first set of experiments concludes that for 10 out of 14 f-series instances and 9 out of 11 problem instances, the proposed solution has performed better.

### 5.2.2. No forced recentre

Table 6 displays the results on 25 problem instances without force recentre approach with both representations. The summary result presents the best fitness value for 5 out of 11 problem instances. However, for f-series, it has best value for 7 out of 14 instances. While for contig, 3 and 1 problem instances represent the minimum and same number of contigs. For f-series, 2 and 1, the number of instances show minimum and same number of contigs sequentially. Therefore, the conclusion for without force recentre approach shows that for 9 out of 11 problem instances and 10 out of 14 f-series instances perform better using the proposed solution.

## 5.3. Experiments with PALS for optimizing after executing GA

The second set of experiment repeats the first set of experiments. However, in this experiment instead of using PALS as an evolutionary operator, it is used after the execution of GA. The reason of this is to save time by using it only for optimizing the obtained solution.

**Table 7**
Second set of results with force recentre.

| | Direct | | | Indirect | | |
|---|---|---|---|---|---|---|
| Dataset | Fitness value | Contig | Time | Fitness value | Contig | Time |
| m15421_5 | **21851** | **-10** | 1526771880 | 20579 | -4 | 1526794122 |
| m15421_6 | **29330** | **-48** | 1526672492 | 8694 | -9 | 1526686176 |
| m15421_7 | **33188** | **-40** | 1526720750 | 30735 | -22 | 1526841144 |
| x60189_4 | **6947** | **-12** | 1526683631 | 6022 | -4 | 1526645186 |
| x60189_5 | 8629 | -7 | 1526723499 | **8694** | **-9** | 1526686176 |
| x60189_6 | **9527** | **1** | 1526790020 | 8599 | 11 | 1526701831 |
| x60189_7 | **11656** | **-9** | 1526713655 | 10962 | -3 | 1526801702 |
| j02459_7 | **66770** | **-47** | 1527944784 | 65570 | -33 | 1527948346 |
| acin1 | **46972** | **-298** | 1526878711 | 46837 | -298 | 1526875181 |
| bx842596 4 | **123532** | **-53** | 1528117885 | 122435 | -43 | 1528063311 |
| bx842596 7 | **242925** | **-140** | 1530160013 | 239466 | -130 | 1530270189 |
| f25_305 | **2115** | **8** | 1526647579 | 1695 | 12 | 1526572576 |
| f25_400 | 3046 | 6 | 1526678760 | **3373** | **2** | 1526678919 |
| f25_500 | 4560 | 4 | 1526703753 | **4812** | **0** | 1526682992 |
| f50_315 | **5443** | **9** | 1526753246 | 4304 | 13 | 1526797562 |
| f50_412 | **6389** | **9** | 1526691464 | 6121 | 7 | 1526696305 |
| f50_498 | **6896** | **5** | 1526662758 | 8038 | 3 | 1526706263 |
| f100_307 | 4560 | 4 | 1526703753 | **6209** | **51** | 1526609099 |
| f100_415 | 8282 | 57 | 1526662648 | **8282** | **57** | 1526662648 |
| f100_512 | **11116** | **47** | 1527943310 | 10954 | 47 | 1527129950 |
| f508_354 | **44153** | **265** | 1527930106 | 43559 | 263 | 1528361477 |
| f635_350 | 54414 | 324 | 1528006362 | **54924** | **324** | 1528470553 |
| f737_355 | **55900** | **434** | 1527962099 | 55424 | 440 | 1528481359 |
| f1343_354 | 43927 | 265 | 1528472943 | **111687** | **732** | 1530042773 |
| f1577_354 | **122258** | **912** | 1530091905 | 121706 | 916 | 1528069185 |

### 5.3.1. Forced recentre

Table 7 shows the results on 25 datasets for the second set of experiments with the force recentre approach. Result shows the proposed approach achieves best fitness value for 3 out of 11 problem instances. The results on f-series datasets give better fitness value for 3 out of 14 instances. However, the f-series shows the same and minimum number of contigs for 5 and 2 instances, respectively. The force recentre for second experiments conclude that best fitness values are found for 3 out of 11 problem instances. For f-series datasets, the results are better for 10 out of 14 instances.

### 5.3.2. Without forced recentre

Table 8 shows the results without force recentre for the second set of experiments. The summary result shows that none of the problem instance performed better than the past algorithms using the proposed method. However, for f-series datasets, there are only 3 out of 14 instances having better fitness values. Comparing based on contig, there is 1 and 3 instances for the same and minimum number of contigs. This experiment shows that overall there are only 7 out of 14 instances which perform better.

The comparisons between the first and second set of experiments show that PALS perform better when used as a genetic operator. This is because it improves each individual from generation to generation and at the end gives better solution. The first set of experiments perform better with the force recentre and without force recentre for 19 instances out of 25. However, for the second set of experiments, the performance gets degraded and it performed better for 13 and 7 instances for force recentre and without force recentre, respectively.

### 5.6. Experiments with swapping GA representation

The third set of experiments combine the first two sets of experiments with swapping representations. For example, if one run of GA is with a direct representation, then the second time it executes with an indirect transposition-based representation and vice versa.

### 5.6.1. With force recentre

Table 9 shows the summary of the third set of experiments for force recentre. The proposal performed better for 1 out of 11 problem instances and 2 out of 14 f-series instances. However, when contig is compared, there are 2 problem

**Table 8**
Second set of results without force recentre.

| | Direct | | | Indirect | | |
|---|---|---|---|---|---|---|
| Dataset | Fitness value | Contig | Time | Fitness value | Contig | Time |
| m15421_5 | 8373 | -7 | 1527218598 | **8465** | **-7** | 1527044228 |
| m15421_6 | **28275** | **-32** | 1527910382 | 27482 | -30 | 1527918138 |
| m15421_7 | **32157** | **-24** | 1527289281 | 31708 | -22 | 1527289935 |
| x60189_4 | **6200** | **-4** | 1527038783 | 6142 | -4 | 1527037439 |
| x60189_5 | **9612** | **-1** | 1527047678 | 8803 | -9 | 1527048804 |
| x60189_6 | **9032** | **3** | 1527054770 | 8328 | 7 | 1527057723 |
| x60189_7 | **11009** | **-1** | 1527019717 | 10998 | 3 | 1527062885 |
| j02459_7 | **65570** | **-33** | 1527948346 | 66437 | -45 | 1527982218 |
| acin1 | 46978 | -298 | 1527160100 | **47005** | **-298** | 1527160100 |
| bx842596 4 | **123249** | **-53** | 1530219318 | 122186 | -45 | 1530218956 |
| bx842596 7 | **242616** | **-144** | 1530327389 | 241079 | -134 | 1530501848 |
| f25_305 | 1644 | 8 | 1527024700 | **1904** | **8** | 1527031152 |
| f25_400 | **3511** | **4** | 1527031717 | 3190 | 2 | 1527028797 |
| f25_500 | **4818** | **0** | 1527040259 | 4444 | 2 | 1526994686 |
| f50_315 | **4070** | **17** | 1527035577 | 3805 | 19 | 1527035562 |
| f50_412 | **6029** | **13** | 1527048192 | 5812 | 15 | 1527055023 |
| f50_498 | **8432** | **1** | 1527064390 | 7637 | 7 | 1527020713 |
| f100_307 | 6077 | 53 | 1527065330 | **6119** | **53** | 1527064790 |
| f100_415 | 8132 | 57 | 1527106847 | **8159** | **55** | 1527108865 |
| f100_512 | 11402 | 47 | 1527156264 | **11458** | **47** | 1527151569 |
| f508_354 | **43927** | **265** | 1528472943 | 43559 | 267 | 1528361477 |
| f635_350 | **54931** | **322** | 1530163236 | 54715 | 324 | 1528387280 |
| f737_355 | **55341** | **440** | 1528519172 | 55286 | 442 | 1528521143 |
| f1343_354 | **112062** | **728** | 1530149150 | 111737 | 732 | 1530131189 |
| f1577_354 | **121755** | **914** | 1530300770 | 121565 | 916 | 1530194222 |

instances having the same number of the contigs. For the f-series, there are 4 and 1 instance having minimum and same number of contigs. These experiments conclude that there are 3 out of 11 problem instances that are having better values while for f-series instance, there are 7 out of 14 instances having a better value.

### 5.6.2. Without force recentre

Table 9 also shows the summary of the third set of experiments without force recentre. There are 3 out of 11 problem instances that show comparatively better values. The results for f-series datasets represent better value for 2 out of 14 instances. While comparing the number of contigs, there is 1 problem instance having the same value. However, for f-series datasets, it shows better value for 3 out of 14 instances. The third set of experiments without force recentre perform better for 4 out of 11 problem instances and 5 out of 14 f-series instances.

The comparisons of all the three sets of experiments conclude that the proposed method perform well when PALS use it as a genetic operator. In the third set of experiments the performance is degraded due to the representation swapping. The results for force recentre act better for 10 out of 25 datasets. However, without force recentre the results for 9 out of 25 f-series instances perform best. The second set of experiments perform best for 13 and 7 instances for force and without force recentre approach, respectively. However, the first set of experiments show better results for 19 instances using force and without force recentre approach.

In the present work, overlap scores are used as a fitness value. Based on this, individuals with better fitness value will pass on to the next generation. Therefore, only those individuals will survive that have better overlap score. A convergence graph of the proposed work using x601896 dataset is shown in Fig. 8.

**Table 9**
Third set of experiments on all datasets.

| | Force recentre | | | Without force recentre | | |
|---|---|---|---|---|---|---|
| Datasets | Fitness value | Contig | Time | Fitness value | Contigs | Time |
| m15421_5 | 21170 | -12 | 1527908616 | **21974** | **-8** | 1527919051 |
| m15421_6 | 28295 | -36 | 1527908653 | **28493** | **-40** | 1527916198 |
| m15421_7 | 31680 | -26 | 1527871612 | **32115** | **-30** | 1527927321 |
| x60189_4 | 6413 | -8 | 1527896556 | **6484** | **-6** | 1527896540 |
| x60189_5 | 8252 | -5 | 1527897698 | **8454** | **-7** | 1528419149 |
| x60189_6 | **9343** | **1** | 1527898426 | 8855 | 3 | 1527899167 |
| x60189_7 | 10906 | 3 | 1527900576 | **11833** | **-5** | 1527906268 |
| j02459_7 | **68283** | **-51** | 1527068914 | 66323 | -39 | 1528445333 |
| acin1 | 46755 | -298 | 1527761450 | **47060** | **-298** | 1527906376 |
| bx842596 4 | **124304** | **-55** | 1530044793 | 122064 | -45 | 1530452971 |
| bx842596 7 | **241886** | **-142** | 1530427891 | 42621 | 648 | 1530256206 |
| f25_305 | 1637 | 12 | 1527485726 | **1746** | **10** | 1527891640 |
| f25_400 | **3384** | **4** | 1527852955 | 2767 | 8 | 1527898626 |
| f25_500 | 4605 | 0 | 1527900388 | **5131** | **-2** | 1527900207 |
| f50_315 | 3683 | 19 | 1527895056 | **4128** | **21** | 1527897635 |
| f50_412 | **8622** | **55** | 1527773128 | 5764 | 13 | 1527902963 |
| f50_498 | 6871 | 1 | 1527901366 | **7542** | **7** | 1527907971 |
| f100_307 | **8350** | **49** | 1532464964 | 6184 | 53 | 1532464172 |
| f100_415 | **8585** | **53** | 1527905094 | 8349 | 59 | 1527979247 |
| f100_512 | 10343 | 49 | 1527766992 | **10654** | **49** | 1527926483 |
| f508_354 | 43920 | 263 | 1527980270 | **44044** | **263** | 1528050598 |
| f635_350 | 54861 | 320 | 1527811795 | **54983** | **322** | 1528514660 |
| f737_355 | **56316** | **434** | 1527824846 | 55867 | 438 | 1528015421 |
| f1343_354 | **112653** | **726** | 1528098144 | 111956 | 728 | 1530096617 |
| f1577_354 | **121793** | **916** | 1530163002 | 121496 | 916 | 1530143579 |

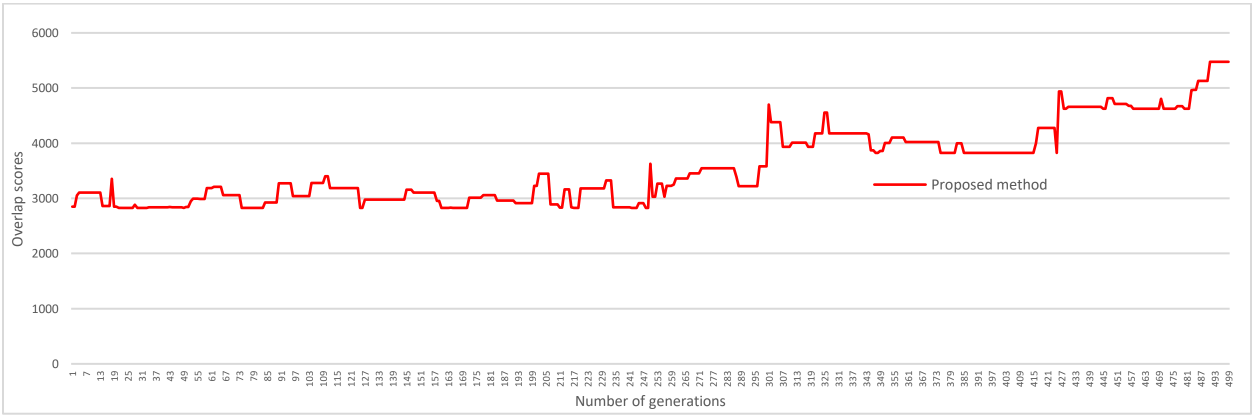


**Fig. 8.** Convergence graph of the proposed approach.

## 5.7. Statistical significance

It is important to see the statistical significance of the obtained results using the proposed approach in comparison to the other competing methods. This significance is identified here using the paired sample *t*-test. First, two null hypothesis ($H_{10}$, $H_{20}$) and their alternative hypothesis ($H_{1A}$, $H_{2A}$) are defined. These are listed in Table 10. Afterwards, the scores according to the number of contigs are assigned to the competing approaches based on their performance over 25 datasets. These scores are listed in Table 11. The proposed work has two objectives, therefore a statistical test is performed for each of these. The significance level $\alpha$ is set to 5% and degree of freedom (*df*) is set to 25. The *p* value is a measure of evidence strength against the null hypothesis that is produced by the sample.

The *t*-statistic is computed for each pair. For this, first scores are assigned to all approaches for each dataset (see Table 11). The score is the ratio of the number of contigs generated by an approach and total number of reads in a dataset. The results in Table 12 show that *t*-statistic was calculated for 25 samples with $\alpha$=0.05 as significance level. As shown in the table there is a significant difference between the proposed work and RRGA [t (24) = 8.7376, $p$< 0.05], GA [t (24) = 72.528755, $p$<0.05], GA+PALS [t (24) = 61.956615, $p$< 0.05], and PALS [t (24) = 8.9909378, $p$< 0.05]. On the basis of the $p$-value which is lower than $\alpha$, the null hypothesis $H1_0$ is rejected in favor of $H1_A$. Therefore, it is concluded that there is a significant difference between the groups.

**Table 10**
Null hypotheses with their alternatives.

| Null hypotheses | Alternate hypothesis |
|---|---|
| $\boldsymbol{H1_0}$ :The proposed technique does not reduce the sum of contigs. | $\boldsymbol{H_{1A}}$ :The proposal reduces the sum of contigs. |
| $\boldsymbol{H2_0}$: The proposed technique does not maximize the sum of overlaps scores. | $\boldsymbol{H_{2A}}$ : The current proposal maximizes the sum of overlaps scores. |

**Table 11**
Score assignment to the competing approaches for first hypothesis.

| Dataset | Proposed | RRGA | GA | GA+PALS | PALS |
|---|---|---|---|---|---|
| m15421_5 | -0.11024 | -0.15748 | 0.708661 | 0.755906 | 2.03937 |
| m15421_6 | -0.2659 | -0.24277 | 0.763006 | 0.855491 | 1.67052 |
| m15421_7 | -0.21469 | -0.24859 | 0.824859 | 0.847458 | 1.463277 |
| x60189_4 | -0.20513 | 0 | 0.051282 | 0.410256 | 7.846154 |
| x60189_5 | -0.22917 | -0.22917 | 0.520833 | 0.479167 | 5.395833 |
| x60189_6 | -0.10606 | -0.16667 | 0.681818 | 0.530303 | 4.045455 |
| x60189_7 | -0.04412 | -0.07353 | 0.338235 | 0.338235 | 3.808824 |
| j02459_7 | -0.16761 | -0.2017 | 0.798295 | 0.815341 | 0.735795 |
| acin1 | -0.97068 | -0.97068 | -0.95114 | -0.97068 | 0.824104 |
| bx842596 4 | -0.14706 | -0.13348 | 0.925339 | 0.929864 | 1.027149 |
| bx842596 7 | -0.20699 | -0.19664 | 0.944373 | -0.38551 | 0.587322 |
| f25_305 | 0.16 | 0.4 | 0.48 | 0.56 | 20.08 |
| f25_400 | 0.16 | 0.16 | 0.24 | 0.48 | 12.08 |
| f25_500 | -0.14815 | -0.14815 | 0.074074 | 0.148148 | 18.59259 |
| f50_315 | 0.38 | 0.34 | 0.46 | 0.58 | 5.54 |
| f50_412 | 0.34 | 0.22 | 0.38 | 0.26 | 7.68 |
| f50_498 | -0.06 | 0.02 | -0.02 | 0.26 | 7.86 |
| f100_307 | 0.43 | 0.43 | 0.65 | 0.85 | 2.95 |
| f100_415 | 0.51 | 0.53 | 0.89 | 0.67 | 3.3 |
| f100_512 | 0.41 | 0.45 | 0.81 | 0.73 | 4.66 |
| f508_354 | 0.517717 | 0.633858 | 0.966535 | 0.927165 | 0.55315 |
| f635_350 | 0.510236 | 0.670866 | 0.951181 | 0.948031 | 0.529134 |
| f737_355 | 0.578019 | 0.985075 | 0.998643 | 0.971506 | 1.002714 |
| f1343_354 | 0.537602 | 0.674609 | 0.99032 | 0.993299 | 0.218913 |
| f1577_354 | 0.58085 | 0.574509 | 0.993025 | 0.994293 | 0.216233 |
| Average | 0.089546 | 0.132802 | 0.578774 | 0.559131 | 4.588262 |
| SD (σ) | 0.382415 | 0.431921 | 0.449868 | 0.458208 | 5.386027 |

**Table 12**
Paired sample *t*-test for first hypothesis.

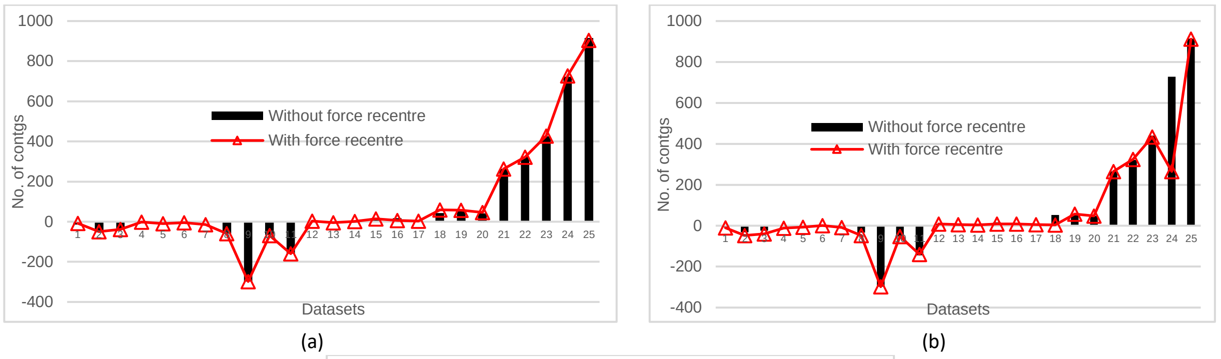
| Pairs | Paired differences: Mean | Std. deviation | Std. error mean | 95% confidence interval of the difference: Lower | Upper | T | df | Sig. (2-tailed) |
|---|---|---|---|---|---|---|---|---|
| Proposed-RRGA | -0.0432566 | -0.0495058 | -0.0099012 | -0.0228206 | -0.0637 | 8.7376 | 24 | 6.40E-09 |
| Proposed-GA | -0.4892281 | -0.067453 | -0.0134906 | -0.4613835 | -0.5171 | 72.528755 | 24 | 1.24E-29 |
| Proposed-GA+PALS | -0.4695852 | -0.0757926 | -0.0151585 | -0.438298 | -0.5009 | 61.956615 | 24 | 5.35E-28 |
| Proposed-PALS | -4.498716 | -5.0036116 | -1.0007223 | -2.4332252 | -6.5642 | 8.9909378 | 24 | 3.76E-09 |

**Table 13**
Scores assigned to the competing approaches for second hypothesis.

| Dataset | Proposed | RRGA | GA | GA+PALS | PALS |
|---|---|---|---|---|---|
| m15421_5 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| m15421_6 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| m15421_7 | 0.75 | 0.75 | 0.5 | 0 | 0.25 |
| x60189_4 | 0.75 | 0.5 | 0.75 | 0 | 0.25 |
| x60189_5 | 0.75 | 0.75 | 0.5 | 0 | 0.25 |
| x60189_6 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| x60189_7 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| j02459_7 | 0.75 | 1 | 0.5 | 0.25 | 0 |
| acin1 | 1 | 1 | 0.5 | 0 | 0.25 |
| bx842596 4 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| bx842596 7 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| f25_305 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| f25_400 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| f25_500 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| f50_315 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| f50_412 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| f50_498 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| f100_307 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| f100_415 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| f100_512 | 1 | 0.75 | 0.5 | 0 | 0.25 |
| f508_354 | 0.75 | 0.75 | 0.5 | 0 | 0.25 |
| f635_350 | 0.75 | 1 | 0.5 | 0.25 | 0 |
| f737_355 | 0.75 | 1 | 0.5 | 0 | 0.25 |
| f1343_354 | 0.75 | 1 | 0.5 | 0.25 | 0 |
| f1577_354 | 1 | 0.75 | 0.5 | 0.25 | 0 |
| Average | 0.86 | 0.85 | 0.51 | 0.04 | 0.21 |
| SD (σ) | 0.126656 | 0.144338 | 0.05 | 0.093541 | 0.093541 |

**Table 14**
Paired sample *t*-test for second hypothesis.

| Pairs | Paired differences: Mean | Std. deviation | Std. error mean | 95% confidence interval of the difference: Lower | Upper | T | df | Sig. (2-tailed) |
|---|---|---|---|---|---|---|---|---|
| Proposed-RRGA | 0.01 | -0.01768 | -0.0035364 | 0.017285 | 0.002715 | -28.27756 | 24 | 6.12E-20 |
| Proposed-GA | 0.35 | 0.076656 | 0.0153311 | 0.318418 | 0.381582 | 45.658704 | 24 | 7.65E-25 |
| Proposed-GA+PALS | 0.82 | 0.033114 | 0.0066229 | 0.806357 | 0.833643 | 247.6274 | 24 | 2.07E-42 |
| Proposed-PALS | 0.65 | 0.0331143 | 0.0066229 | 0.636357 | 0.663643 | 196.29002 | 24 | 5.46E-40 |



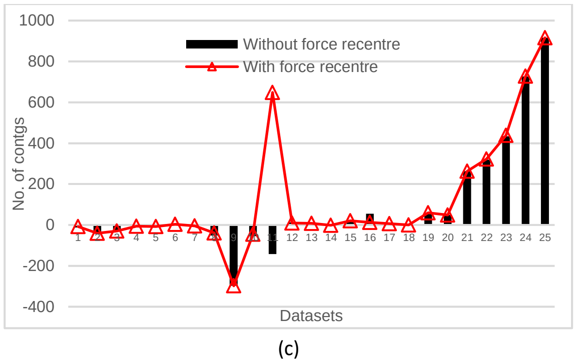
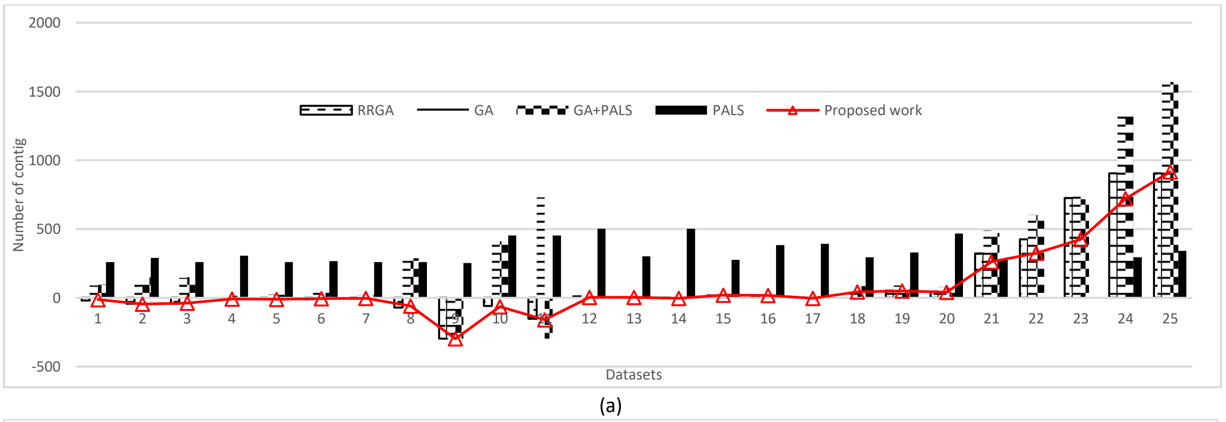


**Fig. 9.** Analysis of the conducted experiments on the 25 benchmark datasets. (a) Initial fragment position optimized through 2-opt, (b) PALS for optimization after executing GA (c) swapping GA representation.



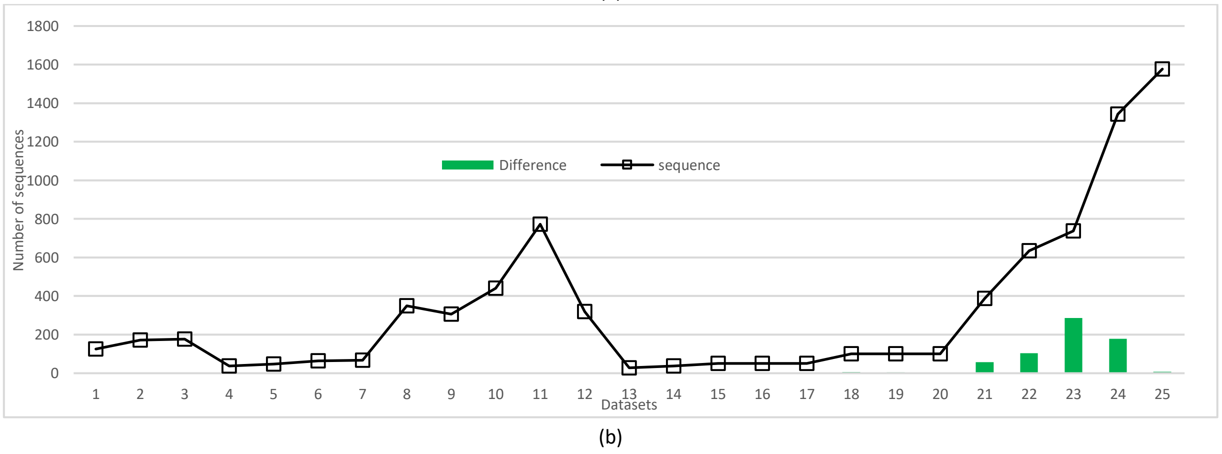


**Fig. 10.** Performance summary of the five competing methods. (a) proposed approach vs. other state-of-the-art methods (b) proposed approach vs. RRGA.

The second hypothesis is about the supposition that the proposed tool does not maximize the overlap scores. For this analysis the score is assigned by comparing the overlap scores among the five approaches. If the proposed approach's overlap score is greater than all the competing algorithms then it is assigned a score of 4 out of 4. These scores are listed in Table 13. The paired sample *t*-test for this analysis is shown in Table 14 which shows that there is a significant difference between the proposed work and RRGA [t (24) = -28.27756, $p< 0.05$], GA [t (24) = 45.658704, $p<0.05$], GA+PALS [t (24) = 247.6274, $p< 0.05$], and PALS [t (24) = 196.29002, $p< 0.05$]. On the basis of the *p*-value which is lower than $\alpha$, the null hypothesis $H2_0$ is rejected in favor of $H2_A$. It is therefore concluded that there is a significant difference between the groups.

The statistical analysis here proves that the proposed work assemble the fragments based on maximizing the number of overlap scores and minimizing the number of contigs. It also suggests that the RRGA [t (24) = 8.7376, $p< 0.05$] and RRGA [t (24) = -28.27756, $p< 0.05$] perform better than the other three comparison methods. This is the primary reason of comparing results of the proposed approach with RRGA.

## 6. Discussion

This work focused on the DNA-FAP and presented a soft computing-based solution. The DNA-FAP is concerned with the reconstruction of the target DNA. For this, it identifies the right order and orientation of nucleotides for each fragment in the layout by taking a vast number of sequenced fragments into consideration. The proposed solution presented a bi-objective optimization problem addressed through two main objectives. The objectives included (a) maximizing the sum of overlap scores between the two adjacent fragments and (b) minimizing the number of contigs. The current work focused on the OLC approach. For initializing the fragment's position, the TSP heuristic, i.e., 2-opt was used to reset the position of the fragments. The RRHGA was used to explore all feasible solutions. Additionally, this work employed the PALS algorithm as an evolutionary operator to correctly order the fragments (genes) of the chromosome. For performance evaluation, 25 benchmark datasets including the collection of 14 f-series datasets were used. Three sets of experiments were conducted using these datasets utilizing two approaches, i.e., with force recentre and without force recentre. The current work was evaluated using three different types of tests. All these tests were carried out with force and without force recenter for both types of representation. The present proposal was compared with RRGA variation for the DNA fragment assembly, PALS, GA, and hybrid GA on the bases of the number of contigs. The main objective of this work was to maximize the sum of overlap scores while maintaining the lowest number of contigs. In the following, experiment wise discussion on the obtained results is presented.

In the first set of experiments, the use of PALS as a genetic operator revealed that when the force recenter is used for the direct representation, the current methodology performed better in 19 out of 25 datasets. However, without force recenter and with direct representation, it performed better for 21 out of 25 datasets. Therefore, it can be concluded that the proposed approach performs better for the first set of experiments without force recenter and with the direct representation in case of minimizing the number of contigs. This observation can be seen in Fig. 9(a).

The second set of experiments represented the effects of PALS algorithm by applying it to the solution obtained by the GA. These types of experiments were conducted with and without force recenter for both types of representation. The proposed methodology performed better in both the cases for the direct representation. However, comparing the experiments with and without force recenter, it performed better for 20 and 23 out of 25 datasets, respectively. Thus, it can be concluded that this methodology scores well without force recenter with direct representation as shown in Fig. 9(b). The third set of experiment combined the first two experiments with swapping representation. These experiments performed better for 15 out of 25 datasets without force recenter and scored better for 21 out of 25 datasets in case of force recentre. For these types of experiments, the presented methodology performed better with force recentre (Fig. 9 (c)).

From the analysis presented in Fig. 9, it can be observed that the proposed solution performs better without the force recentre option and with direct representation. It is further observed form the results that the proposed approach performed better on 80% of the datasets in the first set of experiments and it had better performance on 60% of the datasets in the third set of experiments using the force recntre setting. For the setting without force recntre, the current proposal performed better on 84% datasets in the first and third set of experiments. Whereas, for the same setting it performed better in 92% cases during the second set of experiments.

After evaluating the proposed work's performance in isolation, the next step is to analyze it without force recentre option and with the direct representation for all the three experiments. This was done here while comparing the present proposal with the best performing approach among the four state-of-the-art methods used for comparison, i.e., RRGA. In the first experiment, the present proposal performed better for 88% datasets. Whereas, in the second and third set of experiments, the proposed solution performed better for 72% and 68% datasets, respectively. In contrast to the first set of experiments, from these experiments, it can be seen that the proposed methodology performs better without the force recentre option for the direct representation.

Fig. 10(a) shows the performance comparison summary between the proposed method and the four state-of-the-art approaches to solve DFA. Where, the *x*-axis lists the dataset number and *y*-axis mentions the number of contigs extracted by each of the competing methods. Performance comparison with RRGA revealed that the proposed model perform well in term of minimizing the number of cotings when there are large number of sequences in a dataset. The datasets with larger sequences such as f508_354, f635_350, f737_355, and f1343_354 have comparatively high difference. Whereas, on datasets with smaller sequences, it performs either same or better as compared to RRGA as shown in Fig. 10(b). However, for bx842596 4, bx8425967, and f1577_354 datasets, RRGA performed better because the length of the read is large. Based on this discussion, it can be said that when the number of sequences per dataset are larger and the number of bp of the sequence is small, the proposed model reduces the number of cotings while assembling the sequences.

## 7. Conclusion and future directions

This work presented a soft computing-based hybrid approach for Overlap Layout Consensus task in the DNA fragment assembly. The proposal was based on the combination of metaheuristics, like Restarting and Recentering Genetic Algorithm (RRGA) and Power Aware Local Search (PALS) algorithm. Where, PALS was utilized as an evolutionary operator. The goal of fragment assembly is to determine the exact order of the fragments that minimizes the number of contigs in order to get a single contig. Therefore, the objective function utilized in this work was to maximize the overlap scores and decrease the number of contigs. PALS, as a genetic operator, optimized the individuals by finding adjacent fragments based on threshold along with the number of contigs when the tentative solution was created/destroyed. Due to changes in the tentative solution, PALS exploited the neighbors efficiently by computing partial fitness instead of complete fitness and produced better results. The 2-opt heuristic was also utilized with the GA. Advantage of the heuristic approach was taken by using it for generating the initial seeds for the RRGA. Three types of experiments were conducted with and without force recentre option for avoiding the local optima. In the first set of experiments, PALS was used as a genetic operator. The second set of experiments used PALS after executing the GA. Whereas, in the third set of experiments PALS was used as a genetic operator and was also utilized after GA execution. In the experiments, 25 benchmark datasets were used. Fourteen of these datasets were f-series instances. The comparison of the proposed algorithm was made with four state-of-the-art related approaches for the same problem, i.e., RRGA variations for DNA fragment assembly, PALS, GA, and hybrid GA. The results showed, on an average, better performance of the proposed solution. For the setting without force recntre, in the first set of experiment, the present proposal performed better for 88% datasets. Whereas, in the second and third set of experiments, the proposed solution performed better for 72% and 68% datasets, respectively. Performance comparison with RRGA revealed that the proposed model perform well in term of minimizing the number of cotings when there are large number of sequences in a dataset. Whereas, on datasets with smaller sequences it performs either same or better as compared to RRGA. The proposed model reduces the number of cotings while assembling the sequences when the sequences per dataset are larger and the read length is small.

The solution presented in this work can be extended in multiple ways in the future. In future, it is recommended to employ greedy algorithm instead of GA which may substantially decrease the execution time. The population can be generated by transposition representations as well. This will cause the individuals to be better in each iteration by eliminating some parts of the candidate solutions. Recently, deep learning methods have shown promising results for various classification tasks. It will be interesting to see the performance of deep learning for an unsupervised task, like DFA integrated with the GA. One of the performance bottleneck of the current proposal is repetitive assessments of a candidate solution by multiple population-based optimization techniques. In the future, the proposed framework can be transformed into its parallel equivalent using the parallel programming frameworks, like MapReduce. This may result in huge performance gain in terms of execution time.

## Acknowledgement

The authors are indebted to the editor and anonymous reviewers for their helpful comments and suggestions. The authors wish to thank GIK Institute for providing research facilities. This work was sponsored by the GIK Institute graduate research fund under GA-F scheme.

## Compliance with ethical standards

**Conflict of Interest** The authors declare that they have no conflict of interest.

## References

[1] Y. Oguz , I. Guler, A. Erdem, M.F. Mutlu, S. Gumuslu, M. Oktem, N. Bozkurt, and M. Erdem, "The effect of swim-up and gradient sperm preparation techniques on deoxyribonucleic acid (DNA) fragmentation in subfertile patients." Journal of assisted reproduction and genetics, Vol. 35, pp. 1-7, 2018.

[2] Y. Ohguchi, T. Nomura, S. Suzuki, M. Takeda, T. Miyauchi, O. Mizuno, S. Shinkuma, Y. Fujita, O. Nemoto, K . Ono, and W.I. McLean, "Gentamicin-Induced Readthrough and Nonsense-Mediated mRNA Decay of SERPINB7 Nonsense Mutant Transcripts." Journal of Investigative Dermatology, Vol. 138, pp.836-843, 2018.

[3] J.L. Risler, M.O. Delorme, H. Delacroix, and A. Henaut "Amino acid substitutions in structurally related proteins a pattern recognition approach: determination of a new and efficient scoring matrix." Journal of molecular biology, Vol. 204, pp.1019-1029, 1988.

[4] F. Depardieu, V. Mejean, and P. Courvalin, "Competition between VanUG repressor and VanRG activator leads to rheostatic control of vanG vancomycin resistance operon expression." PLoS genetics, 10.1371/journal.pgen.1005170,2015.

[5] J. Di Iulio, I. Bartha, E.H. Wong, H.C. Yu, V. Lavrenko, D. Yang, I. Jung, M.A. Hicks, N. Shah, E.F. Kirkness, and M.M, Fabani," The human noncoding genome defined by genetic diversity", Nature genetics, Vol. 50, No. 3, pp. 333-337, 2018.

[6] A.M. Bolger, H. Poorter, K. Dumschott, M.E. Bolger, D. Arend, S. Osorio, H. Gundlach, K.F. Mayer, M. Lange, U. Scholz, and B. Usadel, "Computational aspects underlying genome to phenome analysis in plants. The Plant Journal." Vol. 97, No. 1, pp.182-198, 2019.

[7] D. Earl, K. Bradnam, J.S. John, A. Darling, D. Lin, J. Fass, H.O.K. Yu, V. Buffalo, D.R. Zerbino, M. Diekhans, and N. Nguyen, " Assemblathon 1: a competitive assessment of de novo short read assembly methods", Genome research, Vol. 21, No. 12, pp.2224-2241,2011.

[8] P.A. Pevzner, H. Tang, and M.S. Waterman," An Eulerian path approach to DNA fragment assembly", Proceedings of the National Academy of Sciences, Vol. 98, No. 17, pp.9748-9753, 2001.

[9] K.W. Huang, J.L. Chen, C.S. Yang, and C.W. Tsai, "A memetic particle swarm optimization algorithm for solving the DNA fragment assembly problem", Neural Computing and Applications, Vol. 26, No. 3, pp. 495-506, 2001.

[10] P. Meksangsouy, N. Chaiyaratana, "DNA fragment assembly using an ant colony system algorithm", In IEEE Evolutionary Computation (CEC'03), pp. 1756-1763, 2003.

[11] R.S .Verma, V .Singh, and S .Kumar, "DNA sequence assembly using particle swarm optimization", International Journal of Computer Applications, Vol. 28, No.10, pp. 0975 – 8887, 2011.

[12] M. Elloumi, and S. Kaabi , "Exact and approximation algorithms for the DNA sequence assembly problem", SCI in Biology and Medicine, pp.8,1999.

[13] F. Sanger, S. Nicklen, and A.R. Coulson, "DNA sequencing with chain-terminating inhibitors", Proceedings of the national academy of sciences, Vol. 74, No. 12, pp. 5463-5467, 1977.

[14] Z. Halim and T. Muhammad, "Quantifying and Optimizing Visualization: An Evolutionary Computing-Based Approach," Information Sciences, Vol. 385, pp. 284-313, 2017.

[15] Z. Halim, M. Waqas, A.R. Baig and Ahmar Rashid, "Efficient Clustering of Large Uncertain Graphs Using Neighborhood Information," International Journal of Approximate Reasoning, Vol. 90, pp. 274-291, 2017.

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

[17] J. A. Hughes, S. Houghten, and D. Ashlock, "Recentering, reanchoring & restarting an evolutionary algorithm." In IEEE, World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 76-83, 2013.

[18] K.B. Ignatov, K.A. Blagodatskikh, D.S. Shcherbo, T.V. Kramarova, Y.A. Monakhova, and V.M. Kramarov, “Fragmentation Through Polymerization (FTP): A new method to fragment DNA for next-generation sequencing,” PloS one, 14(4), p.e0210374, 2019.
[19] M. Rathee, K. Dilip, and R. Rathee, ”DNA Fragment Assembly Using Quantum-Inspired Genetic Algorithm,” In Exploring Critical Approaches of Evolutionary Computation, pp. 80-98, 2019.
[20] A. Tangherloni, S. Spolaor, L. Rundo, M.S. Nobile, P. Cazzaniga, G. Mauri, P. Liò, I. Merelli, and D. Besozzi, “GenHap: a novel computational method based on genetic algorithms for haplotype assembly,” BMC bioinformatics, 20(4), p.172, 2019.
[21] T. Li, T. He, Z. Wang, and Y. Zhang, ”SDF-GA: a service domain feature-oriented approach for manufacturing cloud service composition,” Journal of Intelligent Manufacturing, pp.1-22, 2019.
[22] D. Richter, K. Bayer, T. Toesko, and S. Schuster, ”ZeBRα a universal, multi-fragment DNA-assembly-system with minimal hands-on time requirement,” Scientific reports, 9(1), p.2980, 2019.
[23] G. Raja and U.S. Reddy, ”Maximum Exact Matches for High Throughput Genome Subsequence Assembly,” IETE Journal of Research, pp.1-8, 2019.
[24] J.S. Firoz, M.S. Rahman, and T.K. Saha, “Bee algorithms for solving DNA fragment assembly problem with noisy and noiseless data.", In Proceedings of the 14th annual conference on Genetic and evolutionary computation (ACM), pp. 201-208, 2012
[25] R. Tinós, K. Helsgaun, and D. Whitley, "Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic.", In International Conference on Parallel Problem Solving from Nature, pp. 95-107, 2018.
[26] E. Alba, and G. Luque, “A hybrid genetic algorithm for the DNA fragment assembly problem." In Recent advances in evolutionary computation for combinatorial optimization, pp. 101-112, 2008.
[27] G. Minetti, and E. Alba, "Metaheuristic assemblers of DNA strands: Noiseless and noisy cases." In IEEE Congress on Evolutionary Computation (CEC), pp. 1-8, 2010.
[28] S. Romano, N.X. Vinh, K. Verspoor, and J. Bailey, ”The randomized information coefficient: assessing dependencies in noisy data.” Machine Learning, vol. 107, no. 3, pp. 509-549, 2018.
[29] S. Kikuchi, and G. Chakraborty, "Heuristically tuned GA to solve genome fragment assembly problem." In IEEE Congress on Evolutionary Computation (CEC), pp. 1491-1498, 2006.
[30] J. Hughes, S. Houghten, G.M. Mallen-Fullerton, and D. Ashlock, "Recentering and restarting genetic algorithm variations for DNA fragment assembly." In IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, pp. 1-8, 2014.
[31] G.M. Mallén-Fullerton, and G. Fernandez-Anaya,"DNA fragment assembly using optimization.” In IEEE congress on Evolutionary computation (CEC), pp. 1570-1577, 2013.
[32] A.J. Nebro, G. Luque, F. Luna, and E. Alba, "DNA fragment assembly using a grid-based genetic algorithm”, Computers & Operations Research, Vol. 35, No. 9, pp. 2776-2790, 2008.
[33] A. McCarthy, "Third generation DNA sequencing: pacific biosciences' single molecule real time technology." Chemistry & biology, Vol.17, No.7, pp. 675-676, 2010.
[34] G.F. Minetti, G. Luque, and E. Alba, "The Problem Aware Local Search algorithm: an efficient technique for permutation-based problems.", Soft Computing, Vol. 21, No. 18, pp. 5193-5206, 2017.
[35] T.F. Smith, and M.S. Waterman, "Comparison of biosequences", Advances in applied mathematics, Vol. 2, No. 4, pp. 482-489, 1981.
[36] A.K. Das and D.K. Pratihar, ”Performance improvement of a genetic algorithm using a novel restart strategy with elitism principle,” International Journal of Hybrid Intelligent Systems, 15(1), pp.1-15, 2019.
[37] A.A.R. Hosseinabadi, J. Vahidi, B. Saemi, A.K. Sangaiah, and M. Elhoseny, ”Extended genetic algorithm for solving open-shop scheduling problem,” Soft computing, 23(13), pp.5099-5116, 2019.
[38] H. Lu, R. Zhou, Z. Fei, and C. Guan, ”Spatial-domain fitness landscape analysis for combinatorial optimization,” Information Sciences, 472, pp.126-144,2019.
[39] W.B. Qiao, and J.C. Créput, "Massive 2-opt and 3-opt Moves with High Performance GPU Local Search to Large-Scale Traveling Salesman Problem.", In International Conference on Learning and Intelligent Optimization, pp. 82-97, 2018.
[40] E.D. Taillard and K. Helsgaun, “POPMUSIC for the travelling salesman problem.” European Journal of Operational Research, vol. 272, no. 2, pp.420-429, 2019.

[41] Z. Halim and Uzma, “Optimizing the minimum spanning tree-based extracted clusters using evolution strategy,” Cluster Computing, Vol. 21, No. 01, pp. 377-391, 2018.
[42] R.J. Parsons, S. Forrest, and C. Burks, "Genetic algorithms, operators, and DNA fragment assembly." Machine Learning, Vol. 21, No. 1-2, pp.11-33, 1995.
[43] E. Alba and G. Luque, “A new local search algorithm for the DNA fragment assembly problem.” In European Conference on Evolutionary Computation in Combinatorial Optimization, pp. 1-12, 2007.