# DNA SEQUENCE ASSEMBLY USING PARTICLE SWARM OPTIMIZATION AND NAÏVE CROSSOVER

**SUNIL KUMAR DEWANGAN[a1], GURPREET SINGH CHHABRA[b] AND ASHISH TRIVEDI[c]**

[abc]SSIPMT, Raipur, Chhattisgarh, India

## ABSTRACT

**DNA sequence assembly problem is a very complex problem of computational biology. DNA sequence assembly is NP hard predicament there is absolutely no one solution these kinds of problem. DNA sequence assembly means merging of broken fragments of the much longer DNA sequence so as to reconstruct the original sequence. In this paper we proposed a solution for DNA sequence assembly problem using Particle Swarm Optimization (PSO) with Naïve Crossover and Shortest Position Value (SPV) rule. DNA sequence assembly problem is individually distinct optimization problem, and so there is require regarding individually distinct optimization algorithm in order to resolve that. Within this paper model regarding PSO is used having Naïve Crossover in addition with SPV rule in order to solve the DNA sequence assembly problem. SPV rule changes transforms continuous version of PSO to individually discrete version. Proposed technique is named while DSAPSONC. To evaluate the efficiency regarding proposed technique the final results regarding DSAPSONC is usually in contrast to the final results regarding DNA sequence assembly problem applying Particle Swarm Optimization (DSAPSO).**

**KEYWORDS:** DNA, PSO, Naïve Crossover, PSONC, DSAPSO**.**

The current challenge in the field of biology is the enormous amount of existing data. This data is complex and unformatted. Also this data is doubled in every two years. The bioinformatics is the interdisciplinary research area of biology & computer science. It uses the computer science methods, models and sophisticated algorithms to solve the biological problems that are related to huge data analysis, gene annotation, pattern reorganization and many more.

DNA sequence assembly problem is quite difficult problem along with get much more computational time for obtaining consensus sequence. DNA sequence assembly problem is NP hard problem since there are several solutions available for this sort of problem. Many literatures provide solutions for DNA sequence assembly problem. Alfredia Burks [4] is commonly used by that DNA sequencing throughput must be improved by means of order placed associated with degree to perform the position inside the period of time associated with 15 decades that has been spelled out with the Human Genome Project, and that this kind of stunning raises will certainly really rely in big aspect with automating the several fresh along with interpretive steps involved in DNA sequencing. Over the past few years quite a few fragment assembly deals are already created along with used to sequence different creatures. The most used deals tend to be PHRAP [5] is really a plan pertaining to assembling shotgun DNA sequence info. TIGR assembler [6] overcomes various significant obstructions to help assembling DNA sequence. WANDER [7] implemented a reliable strategy to sequence DNA using primer going for walks method. CAP3 [8] consists of quite a few changes along with brand new functions to enhance DNA sequence assembly. Celera assembler [9] created from Celera with the 2001 guide on the very first write individual genome sequence. EULER [10] will be an approach to fragment assembly that will abandons the particular established "overlap : structure : consensus" paradigm that is certainly employed in many now available assembly tools. Allex, C. F., Baldwin, Utes. F., Sbavlik, T. W. along with Blamer, F. Third. [11] will be bettering the caliber of automatic DNA sequence assembly using neon trace-data classifications. Wilks, C. along with Khuri. [12] proposed " A Structured Pattern Matching Approach to Shotgun Sequence Assembly (AMASS) created by Sun Kim.

Quite a few heuristic solutions are generally used throughout DNA Sequence Assembly that are much better accomplishing this connected with DNA Sequence Assembly one too will be Genetic algorithm. Parsons, 3rd r. and Forrest, Ohydrates. and Burks, C. [13] provides how the Genetic algorithm can be a encouraging way for fragment putting your unit together difficulties, reaching operational answers speedily. Parsons, r. N. and Forrest, Ohydrates. and Burks, C. [14] analyze various innate criteria staff for starters permutation problem for this Individual Genome Project—the putting your unit together connected with DNA sequence fragments at a father or mother identical copy in whose sequence will be unfamiliar in to a agreement sequence related for the father or mother sequence. Parsons, 3rd r. N. and Johnson, M. At the. [15] talked about the newest effects, this improvements for the prior innate criteria utilized, this experimental pattern method where the newest effects were being acquired, this questions lifted simply by these kind of effects, and many preliminary endeavors to explain these kind of effects. Ellie, Okay. and Mohan, CK [16] provides the fragment assembler by using a brand-new

[1]**Corresponding author**

parallel hierarchical adaptive variant connected with evolutionary algorithms. This modern attributes will include a brand-new determine pertaining to analyzing sequence putting your unit together quality and also the growth of an hybrid criteria. Fang, Ohydrates. Chemical. and Wang, Y simply. and Zhong, N. [17] approach maximizes this similarity (overlaps) among given fragments along with a candidate sequence. It thinks the two entire fragments and also the individual basepair resemblances within the sequence. Special innate staff are designed to increase this looking method. Kikuchi, Ohydrates. and Chakraborty, Gary. [18] included a couple of heuristic ideas along with GA for making this better. The first is chromosome decrease (CRed) action that reduce the length of this chromosomes, doing innate search, to boost this productivity. The opposite will be chromosome is purified (CRef) action the industry money grabbing heuristics, rearranging this pieces employing sector knowledge, to locally enhance the fitness connected with chromosomes. Luque, Gary. and Alba, At the. [19] provide various approaches, the canonical innate criteria, the CHC approach, the spread search criteria, along with a simulated annealing, to solve correctly problem cases which have been 77K base pairs lengthy. Meksangsouy, P. and Chaiyaratana, D. [20] proposed the asymmetric placing your order portrayal the place where a path co-operatively generated simply by many ants within the colony symbolizes this search alternative. Zhao, Y simply. and Ma, P. and Lan, N. and Liang, Chemical. and Ji, Gary. [21] much better sequence alignment approach good ould like colony criteria. The brand new approach can prevent a neighborhood the best and take out especially this walkways results connected with fantastic difference simply by controlling the first and ultimate jobs connected with ants and simply by editing pheromones in numerous periods.

There are find few literatures available that signify alternative pertaining to DNA Sequence Assembly problem employing metaheuristic and characteristics encouraged algorithms. PSO criteria will come below characteristics encouraged criteria in fact it is incredibly powerful way to resolve the seo problem. It has been established which PSO handles Search engine optimization problem successfully and provide this the best end result. Cross PSO criteria can give far better end result compared to traditional PSO criteria. DNA sequence putting your unit together problem will be sorted previously simply by PSO criteria that's the reason why we've goal to apply PSO criteria along with Naïve crossover pertaining to DNA Sequence Assembly problem.

## DNA SEQUENCE ASSEMBLY PROBLEM

Deoxyribonucleic acid (DNA) is a nucleic acid that contains the particular ancestral guidance found in the particular advancement and also functioning coming from all identified known organisms and many worms. The main function of DNA elements within living organisms t would be the long-term storage of details.

DNA series is displayed by way of sequence of four-letter alphabet (A, C, G, and T) matching towards the number of monomeric basics that the particular DNA polymer is made up. A portion, or even fragment, goes along in your circumstance to your substring of 100-1000 basics. Overlap power and also offset human relationships among twos of pieces, accustomed to get the particular construction in the pieces in a global layout, is founded on comparison of character guitar strings. The particular production generated through sequencing presents any agreement for the order of 1000-1, 000, 000 basics long, generated through voting within arranged articles of basics resulting from the particular page layout.

The particular construction difficulty is a combinatorial optimization difficulty exactly where the aim of the particular seek is to obtain the right order and also orientation of every fragment from the fragment buying series leading towards the formation of your agreement series.

Figure 1, shows the basic DNA sequence assembly process. Figure 1 represents 4 different DNA fragment taken from large human DNA sequence STIM1. STIM1 DNA sequence is taken from NCBI. After getting 4 fragment from large DNA sequence file arrangement of fragments is needed to calculate consensus sequence.

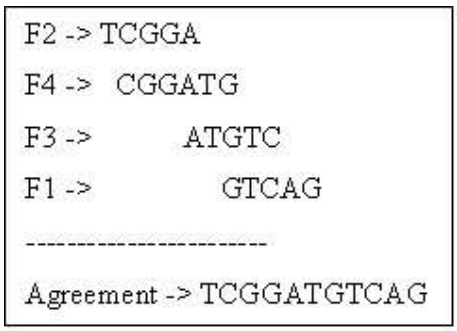


**Figure 1: DNA sequence assemby process**

## PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique for the solution of continuous optimization problems. It is inspired by social behaviors in flocks of birds and

schools of fish [5][6]. In PSO, a set of agents called particles will search for good solutions to a given continuous optimization problem. PSO has been applied in many different problems and has successfully solved this problem better than other algorithms.

The particle swarm optimization algorithm, originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart [1], solves problems in many fields, especially engineering and computer science. The power of the technique is its fairly simple computations and sharing of information within the algorithm as it derives its internal communications from the social behavior of individuals. The individuals, called particles henceforth, are flown through the multi-dimensional search space with each particle representing a possible solution to the multi-dimensional optimization problem. Each solution's fitness is based on a performance function related to the optimization problem being solved.

The movement of the particles is influenced by two factors using information from iteration-to-iteration as well as particle-to particle [2]. As a result of iteration-to-iteration information, the particle stores in its memory the best solution visited so far, called pbest, and experiences an attraction towards this solution as it traverses through the solution search space. As a result of the particle-to-particle information, the particle stores in its memory the best solution visited by any particle, and experiences an attraction towards this solution, called gbest, as well. The first and second factors are called cognitive and social components, respectively. After iteration, the pbest and gbest are updated for each particle if a better or more dominating solution (in terms of fitness) is found. This process continues, iteratively, until either the desired result is converged upon, or it's determined that an acceptable solution cannot be found within computational limits.

For an n-dimensional search space, the i-th particle of the swarm is represented by a n- dimensional vector, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$. The velocity of this particle is represented by another n dimensional vector $vi = (v_{i1}, v_{i2}, \ldots, v_{in})^T$. The previously best visited position of the i-th particle is denoted as $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^T$. 'g' is the index of the best particle in the swarm. The velocity of the i-th particle is updated using the velocity update equation given by

$$v_{id} = v_{id} + c_1 r_1( p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id}), \quad (1)$$

and the position is updated using

$$x_{id} = x_{id} + v_{id} \quad (2)$$

Where d = 1, 2… n represents the dimension and i = 1, 2… S represents the particle index. S is the size of the swarm and c1 and c2 are constants, called cognitive and social scaling parameters respectively (usually, c1= c2; r1, r2 are random numbers drawn from a uniform distribution). Eq. (1) and (2) define the classical version of PSO algorithm. A constant, Vmax, was introduced to arbitrarily limit the velocities of the particles and improve the resolution of the search [4]. The maximum velocity Vmax, serves as a constraint to control the global exploration ability of particle swarm. Further, the concept of an inertia weight was developed to better control exploration and exploitation. The motivation was to be able to eliminate the need for Vmax. The inclusion of an inertia weight in the particle swarm optimization algorithm was first reported in the literature [3]. The resulting velocity update equation becomes:

$$v_{id} = w*v_{id} + c_1 r_1( p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id}), \quad (3)$$

Eberhart and Shi, [3] indicate that the optimal strategy is to initially set w to 0.9 and reduce it linearly to 0.4, allowing initial exploration followed by acceleration toward an improved global optimum[14].

**Problems in PSO**

The PSO method is very popular due to its simplicity of implementation and ability to quickly converge to a reasonably good solution [7][8]. But it has some limitations which are:

- Premature convergence
- Tending to get stuck in local optima
- Low solution precision

**Premature convergence**

When an optimization algorithm converged very soon into a solution and stuck into it, then there is no further enhancement in the best solution and this situation is called premature convergence. Generally when PSO is used to solve problems, it faces premature convergence due to which it gives local best solution as its best solution.

**Tending to get stuck in local optima**

This problem is similar to premature convergence problem but solution converges into a local best solution. Therefore, local solution is considered as best solution. For an optimization problem it might be possible to have many local best solutions. If an optimization algorithm stuck into its one of the local best solutions, then this situation is tending to get stuck in local optima.

**Low Solution Precision**

As a result of above two problems convergence and local optima solution of optimal problem tends to be less and low precise. When solving high-dimension functions problem of low solution precision occurs very frequently.

## CROSSOVER OPERATOR

One of the oldest optimization algorithm is genetic algorithm which is based on natural evolution process. The working principle of genetic algorithm is based on natural evolution process or inheritance from different species. Genetic algorithm uses three main operators which are selection, crossover and mutation. Among the three operators crossover operator has greater significance. Crossover is based on inheritance principle, i.e. adapting existing properties or expanding the existing properties. In crossover operations, new solutions are generated using the existing solutions. The concept has been adapted from natural evolution process, i.e. new generation come into existence because of their parents. There are different versions of crossover operators of genetic algorithm. Depending on problem requirement, crossover operators are used. Mainly crossover operators are of two versions or two categories. One is binary version and another is real version. Depending upon the problem domain crossover operator versions are used. This paper presents the use of real version of crossover operator. Crossover operators are very useful tool for exploring solution space. The problem which has its solution based on real domains needs a good exploration technique to explore solution space effectively. Real version of crossover operator explore solution space very effectively.

**Naïve Crossover**

Naive crossover operator [15] produces two off springs from a pair of parents by randomly selecting a cross site between 1 and n, parents solution dimension and replacing the former and latter half of each parent from the cross site. Let ( $X_1^{(1.t)}$ $X_2^{(1,t)}$ $X_3^{(1,t)}$ $X_4^{(1,t)}$ ….. $X_n^{(1,t)}$) and($X_1^{(2.t)}$ $X_2^{(2,t)}$ $X_3^{(2,t)}$ $X_4^{(2,t)}$ ….. $X_n^{(1,t)}$) are two parent solutions of dimension n at generation t. A cross site of value 3 will produce the offsprings shown in equation (5) and (6).

Offspring 1: ( $X_1^{(1.t)}$ $X_2^{(1,t)}$ $X_3^{(1,t)}$ $X_4^{(2,t)}$ ….. $X_n^{(2,t)}$) (4)

Offspring 2: ( $X_1^{(2.t)}$ $X_2^{(2,t)}$ $X_3^{(2,t)}$ $X_4^{(1,t)}$ ….. $X_n^{(1,t)}$) (5)

## METHODOLOGY

In this paper we now have offered a fix for DNA sequence assembly problem making use of particle swarm optimization with Naïve crossover. Pertaining to dealing with almost any optimization problem we must initial formulate the condition as outlined by optimization problem. In this case initialy we formulate the DNA sequence assembly problem as outlined by PSO criteria. The next subsection describes how we formulate your DNA sequence assembly problem.

To fix the condition, representation from the particular person and also physical fitness value is essential. PSO criteria will be based upon population (candidate solution) and also just about every population get its own physical fitness value as outlined by which often it is in contrast coming from others, consequently we must initial characterize your DNA sequence assembly problem with regards to PSO criteria.

**Individual representation**

Within DNA sequence assembly problem inputs include the set of pieces which often should be constructed and also create a typical sequence which often doesn't need almost any replicated design. The most popular sequence is recognized as because result for DNA sequence assembly problem. To discover your purpose or maybe property involving particular gene history, your examining involving nucleotide or maybe chemical substance basic (A, T, C, G) sequence is completed. Big nucleotide sequences tend to be known as DNA sequence. The particular large DNA sequence is made of replicated behaviour involving nucleotide that's why your DNA sequence turns into large. Within DNA sequence assembly replicated behaviour tend to be eliminated then one opinion sequence creates.

Within DNA sequence assembly large DNA sequence of a particular gene is actually obtained for assembly procedure. Big DNA file is actually divide at random in different pieces involving DNA sequence which can be found in assembly procedure. Soon after obtaining set of fragment, pieces tend to be aimed and the best match between the suffix of merely one sequence and the prefix involving yet another is actually decide. Many feasible match mix off pieces is actually in contrast and also related rating is determined. On such basis as related rating fragment purchase is determined. Eventually your opinion sequence is found out of the fragment purchase. We have done tests using the nucleotide sequences involving homosapiens(human) and mouse viz. MACF1, TNFRSF19 and also Zfa. The particular DNA data is got from NCBI [22].

We have solved DNA sequence assembly problem using the continuous version of PSO. In DNA sequence assembly problem fragment order in which

fragments are aligned is very important but very hard to find the best order from the large possible combinations of fragment order. Using PSO we determine the fragment order. PSO is based on the concept of population and each individual represents a solution for a problem. In case of DNA sequence assembly problem each individual of PSO represents the fragment order on which fragments are aligned to find out a consensus sequence. Each individual has certain dimension value for DNA sequence assembly problem each individual has a dimension value equal to the number of fragments taken for assembly.

DNA sequence assembly problem is a discrete optimization problem. In the proposed solution continuous version of PSO is used instead of discrete version. To change the continuous version to real version for DNA sequence assembly problem SPV rule is used. Using the SPV (shortest position value) rule continuous position generated by PSO is converted to discrete value.

Each individual or particle of PSO is represented by a Position vector $X_{id}$= { $x_1$,$x_2$,$x_3$,.....$x_d$ } where i is the particular individual and d represents the dimension index. Each individual of PSO contain the real value for a particular dimension and on the basis of this real values new sequence vector is generated using shortest position value rule (SPV). New generated sequencevector using SPV is represented as $S_{id}$= [$f_{i1}$,$f_{i2}$,…….$f_{id}$]. $S_{id}$ is a fragment order of i particle in the processing order containing d dimension and $f_{i1}$, $f_{i2}$ represent the fragment number in a fragment order.

For example the individual generated by PSO is $X_{id}$= {4.83, -.55, 1.90, 4.46, 1.05, 2.47, -1.28, 0.192, 3.56, 2.28} which has dimension value equal to 10 that means the number of fragments taken is 10. It is clear that $X_{id}$ contains the real values and for DNA sequence assembly we need fragment sequence order from the set of possible combination of fragment order. SPV rule is used to generate the new sequence vector $S_{id}$. Dimension values of $X_{id}$ is used to generate sequence vector, the dimension index which has the shortest value in $X_{id}$ represents the first fragment that is $f_0$, second shortest value represents the second fragment and so on. The sequence vector $S_{id}$ generated for $X_{id}$ using SPV is {9, 1, 4, 8, 3, 6, 0, 2, 7, 5} here 9 represents the fragment 10 and 1 represents the fragment 2 and so on. $S_{id}$ represents the fragment order in which fragments are aligned for determining the consensus sequence. For each individual of PSO, sequence vector is calculated using the SPV rule.

**Fitness Function**

After representation of each individual we have to calculate fitness value of each individual. On the basis of fitness value we determine the optimal solution. In case of DNA sequence assembly problem optimal solution is the maximum matching score of fragment order.

First we have to align the fragments according to the fragment order $S_{id}$ then longest match between the suffix of one fragment and the prefix of another is determine. Matching score is calculated by counting the matching nucleotide of fragments. The matching score for a pair of fragment is calculated using eq. (6)

$$score_{i,i+1} = \begin{cases} 0, & if\ nucleotide\ does\ not\ matched \\ score_{i,i+1} + 1, & otherwise \end{cases} \quad (6)$$

In eq. (6) $score_{i,i+1}$ is a matching score of two consecutive fragments of sequence vector $S_{id,}$ i and i+1 is the index of sequence vector $S_{id}$. After calculating the score of fragment pair total score is calculated for a particular individual of PSO. Total score is calculated by eq. (7).

$$max\ f_i(x) = \sum_{j=0}^{n-1} score_{j,j+1} \quad (7)$$

In eq. (8) $f_i(x)$ denotes the fitness value for individual i of PSO. In eq. (5) max denotes that our objective is to maximize the value of $f_i(x)$. Individual who has the maximum value of $f_i(x)$ is considered as optimum solution. Fitness function is the summation of all scores calculated by eq. (7) for an individual.

DSAPSONC: DNA Sequence Assembly using PSO Algorithm with Naïve crossover

Conventional PSO has been proved very effective and good problem solving tool. PSO is being used on many optimization problems and PSO has come on modified to various versions. Though PSO is very good tool for optimization problem solving, it has certain limitations. PSO does not work efficiently with every optimization problems due to its short comings. In this paper, a new version of PSO is described using the properties of crossover operator of Genetic algorithm. Crossover operations are also known for good exploration of solution space. It might be possible to overcome the PSO's premature convergence problem using crossover. This new model is a real version of

PSO which can be used to solve problem based on real version problem.

We have used particle swarm optimization algorithm with Naïve crossover to solve DNA sequence assembly problem. In DNA sequence assembly problem inputs are different number of fragment and output is the common sequence which does not have repeated nucleotides. DNA sequence assembly problem is a discrete optimization problem but we have used real version of particle swarm optimization. Real coded PSO is converted to discrete version using shortest position value (SPV) rule. The problem is first formulated according to PSO algorithm. Each individual of PSO with Naïve crossover represents a solution and has a dimension value. For DNA sequence assembly dimension of PSO with Naïve crossover individual is equal to the number of fragments taken.

PSO with Naïve crossover with SPV works in two phases one is initialization phase and other is PSO update phase. In initialization phase individuals are initialized and in update phase solutions are update and new solutions are generated. SPV rule is used to convert the real coded values to discrete values.

First set of solutions are taken randomly within the search space and the fragment order is calculated using the randomly initiated particle using SPV rule. The fitness value is calculated using the fitness equation and the best solution is noted. Next update of the particles is performed using the PSO update equation and the new fragment order is calculated using the SPV rule. Fitness of updated particles is calculated and the best solutions are noted.

This process runs until the maximum function evaluation reached and the best fragment order is noted on the basis of fitness function of individuals. At last on the basis of fragment order fragments are arranged so that matching nucleotides are removed and common consensus sequence are calculated.

## EXPERIMENT RESULTS AND DISCUSSION

This section describes the experimental setup and result obtained after the experiment. We have taken thee DNA sequences for experiment. For each data set we run DSAPSO 30 times with different function evaluation values. The algorithm is simulated using Visual C++. To check the efficiency of proposed DSAPSO algorithm we compare the result of our proposed algorithm with the results of genetic algorithm.

### Experimental Setup

The proposed model uses the properties of both PSO and crossover, so it is necessary to select appropriate parameters for good result. There are many literatures available which describe the standard parameter settings for PSO. An extensive experiment has been performed to setup the parameter for the proposed PSO with Naïve crossover. Maximum number of function evaluation (MaxEval), we have tested our algorithms for three different function evaluation value 2500, 5000 and 10000. The number of population or candidate solution has been taken10, 15 and 30 for PSO. Dimension of each individual or candidate solution is problem dependent. Value for constants '$c_1$ & $c_2$' has been taken as two. Value for 'w' has been taken 0.9. Crossover operator is based on crossover probability parameter, which has been set by an exhaustive experiment. The value of crossover probabilities varies from 0.1 to 0.9, but for our experimental result we have checked every possible value of crossover. The number of run is taken as 30. Solution space parameters are dependent upon problem. The different benchmark function has different search space.

### Real Data Set Used

We used the three real DNA sequence data set as a benchmark for DNA sequence assembly problem. The three real data sets MACF1, TNFRSF19 and Zfa are taken from the National Center for Biotechnology Information (NCBI) [22]. Table 1 shows the data sets name and size of each data set used. The three data sets correspond to alive beings. More concretely, two of the data sets are from the human and one from the mouse. Moreover, we selected data sets with different number of sequences and with different sizes (nucleotides per sequence) to ensure that our algorithm works with several types of instances.

**Table -1 Real Data Set Used**

| Data Set | Size | Source |
|---|---|---|
| MACF1 | 19626 | Human |
| TNFRSF19 | 4371 | Human |
| Zfa | 3469 | Mouse |

### Analysis or Discussion of Experiment

In this section we analyze the result obtained by our algorithm. Here we have shown the comparison of our technique with genetic algorithm. To test the efficiency of proposed DSAPSONC algorithm we have compared the results of DSAPSONC with DSAPSO.

We have performed several experiments in order to obtain the best configuration for our algorithm. We have compared the fitness value or matching score value evaluated by DSAPSONC and DSAPSO. We have tested results for three different set of DNA sequence.

Different number of fragments is taken to check the efficiency of algorithm. The parameter for genetic algorithm is taken standard and the result calculated by genetic algorithm is compared with the particle swarm optimization. Real version of PSO and genetic algorithm is used to solve DNA sequence assembly problem. The value of crossover probabilities varies from 0.1 to 0.9, but for our experimental result we have checked every possible value of crossover.

**Table -2 Matching Score comparison using dataset MACF1 with Crossover probability 0.2**

| | | Number of Fragments | | | | | |
|---|---|---|---|---|---|---|---|
| | | 10 | | 15 | | 30 | |
| DATA SET | Max Eval | DSA PSO | DSAP SONC | DSA PSO | DSA PSO NC | DSA PSO | DSAP SONC |
| MA CF1 | 2500 | 14 | 13 | 21 | 24 | 32 | 37 |
| | 5000 | 13 | 14 | 24 | 24 | 34 | 36 |
| | 10000 | 14 | 14 | 24 | 27 | 35 | 38 |

**Table 3: Matching Score comparison using dataset TNFRSF19 with Crossover probability 0.2**

| | | Number of Fragments | | | | | |
|---|---|---|---|---|---|---|---|
| | | 10 | | 15 | | 30 | |
| DATA SET | Max Eval | DSA PSO | DSAP SONC | DSA PSO | DSA PSO NC | DSA PSO | DSAP SONC |
| TNF RSF 19 | 2500 | 12 | 12 | 18 | 19 | 72 | 79 |
| | 5000 | 12 | 13 | 18 | 19 | 82 | 79 |
| | 10000 | 12 | 12 | 19 | 20 | 87 | 88 |

**Table 4: Matching Score comparison using dataset Zfa with Crossover probability 0.2**

| | | Number of Fragments | | | | | |
|---|---|---|---|---|---|---|---|
| | | 10 | | 15 | | 30 | |
| DATA SET | Max Eval | DSA PSO | DSAP SONC | DSA PSO | DSA PSO NC | DSA PSO | DSAP SONC |
| Zfa | 2500 | 16 | 16 | 24 | 26 | 130 | 134 |
| | 5000 | 16 | 16 | 29 | 27 | 164 | 147 |
| | 10000 | 16 | 17 | 29 | 30 | 169 | 171 |

It is clear from the table 2, 3 and 4 that PSONC with SPV performs better than the PSO algorithm. The value of crossover probabilities varies from 0.1 to 0.9, but for our experimental result we have checked every possible value of crossover. PSONC with SPV gives the better matching score than the PSO with crossover probability 0.2. We have performed experiments with different real DNA sequence found by NCBI and table 2, 3 and 4 represents the solution for the three real DNA data (MACF1, TNFRSF19 and Zfa). Our algorithm performs better for every DNA data than the GA. It is also clear from the tables that as the function evaluation increase the matching scores are also increased.

## CONCLUTION & FUTURE WORK

It can be concluded from the above results that the DSAPSONC is very effective in finding the solution to the DNA Sequence Assembly problem. We have adjusted all the parameters to obtain the best configuration of the algorithm for this problem. We have used three different types of real data set to ensure the effectiveness of our algorithm. The data sets are taken from National Center for Biotechnology Information (NCBI). First the PSONC generates the random solution from the search space and each individual contains the real values. Problem of DNA sequence assembly is a discrete optimization problem so value generated by PSONC is changed to the discrete for changing the real value to discrete SPV rule is used. PSO updates its solution in each iteration and new real value generated. After modification of PSONC individual, SPV is used to generate new fragment order to arrange the fragments for calculating the matching scores. Fitness values are calculated for the entire updated individual. This process continues till the maximum number of function evaluation reached. At last the global best fragment order is consider for the calculation of consensus sequence. Then results are compared with the results of DSAPSO.

In future we have intension to apply various nature inspired algorithms for DNA sequence assembly problem and compare their results with our proposed DSAPSONC algorithms result. Hybridization of algorithm may give the better result than the previous existing algorithms so we also want to hybridize our proposed algorithm with some other existing algorithms. PSO is present in various variants so we can also try to apply PSO variants to solve the DNA sequence assembly problem.

## REFERENCES

Kennedy, J. and Eberhart, R. ―Particle swarm optimization‖, Neural Networks, 1995. Proceedings. IEEE International Conference on, Vol. 4, pp. 1942—1948, 1995.

Shi, Y. and Eberhart, R. ―A modified particle swarm optimizer‖, Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on, pp. 69—73, 1998.

Eberhart, R.C. and Shi, Y. ―Comparing inertia weights and constriction factors in particle swarm

optimization‖, Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, Vol. 1, pp. 80—84, 2000.

Burks, C. ―DNA sequence assembly‖, Engineering in Medicine and Biology Magazine, IEEE, Vol. 13, pp. 771—773, 1994.

P. Green. Phrap. http://www.phrap.org/.

G.G. Sutton, O. White, M.D. Adams, and A.R. Kerlavage. ―TIGR Assembler: A new tool for assembling large shotgun sequencing projects‖, Genome Science & Tech., Vol. 1, pp. 9–19, 1995.

T. Chen and S. Skiena. ―Trie-based data structures for sequence assembly‖, Combinatorial Pattern Matching, pp. 206–223, 1998.

X. Huang and A. Madan. ―CAP3: A DNA sequence assembly program‖, Genome Research, Vol. 9, pp. 868– 877, 1999.

E.W. Myers. ―Towards simplifying and accurately formulating fragment assembly‖, Journal of Computational Biology, Vol. 2, pp. 275–290, 2000.

P.A. Pevzner. ―Computational molecular biology: An algorithmic approach‖, The MIT Press, Vol. 1, 2000.

Allex, C.F. and Baldwin, S.F. and Shavlik, J.W. and Blattner, F.R. ―Improving the quality of automatic DNA sequence assembly using fluorescent trace-data classifications‖, Proceedings, Fourth International Conference on Intelligent Systems for Molecular Biology, pp. 3—14, 1996.

Wilks, C. and Khuri, S. ―A fast shotgun assembly heuristic‖, Computational Systems Bioinformatics Conference, Workshops and Poster Abstracts, IEEE, pp. 122—123, 2005.

Parsons, R. and Forrest, S. and Burks, C. ―Genetic algorithms for DNA sequence assembly‖, Proceedings of the First International Conference on Intelligent Systems for Molecular Biology (ISMB-93), pp. 310—318, 1993.

Parsons, R.J. and Forrest, S. and Burks, C. ―Genetic algorithms, operators, and DNA fragment assembly‖, Machine Learning, Vol. 21, pp. 11—33, 1995.

Parsons, R.J. and Johnson, M.E. ―DNA sequence assembly and genetic algorithms--new results and puzzling insights‖, Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (ISMB-95), pp. 277—284, 1995.

Kim, K. and Mohan, CK ―Parallel hierarchical adaptive genetic algorithm for fragment assembly‖, Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, Vol. 1, pp. 600—607, 2003.

Fang, S.C. and Wang, Y. and Zhong, J. ―A Genetic Algorithm Approach to Solving DNA Fragment Assembly Problem‖, Journal of Computational and Theoretical Nanoscience, Vol. 2, pp. 499—505, 2005.

Kikuchi, S. and Chakraborty, G. ―Heuristically tuned GA to solve genome fragment assembly problem‖, Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pp. 1491—1498, 2006.

Luque, G. and Alba, E. ―Metaheuristics for the DNA fragment assembly problem‖, International Journal of Computational Intelligence Research, Vol. 1, pp. 98—108, 2005.

Meksangsouy, P. and Chaiyaratana, N. ―DNA fragment assembly using an ant colony system algorithm, Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, Vol. 3, pp. 1756—1763, 2003.

Zhao, Y. and Ma, P. and Lan, J. and Liang, C. and Ji, G. ―An Improved Ant Colony Algorithm for DNA Sequence Alignment‖, 2008 International Symposium on Information Science and Engineering, pp. 683—688, 2008.

NCBI- http://www.ncbi.nlm.nih.gov/nuccore