DNA Sequence Assembly using Particle Swarm Optimization

Ravi Shankar Verma
National Institute of Technology
Raipur, India

Vikas Singh
ABV- Indian Institute of Information Technology and management, Gwalior, India

Sanjay Kumar
National Institute of Technology
Raipur, India

ABSTRACT
DNA sequence assembly problem is a very complex problem of computational biology. DNA sequence assembly is a NP hard problem there is no single solution available for this kind of problems. DNA sequence assembly refers to aligning and merging fragments of a much longer DNA sequence in order to reconstruct the original sequence. In this paper a solution is proposed for DNA sequence assembly problem using Particle Swarm Optimization (PSO) with Shortest Position Value (SPV) rule. DNA sequence assembly problem is a discrete optimization problem, so there is need of discrete optimization algorithm to solve it. In this paper continuous version of PSO is used with SPV rule to solve the DNA sequence assembly problem. SPV rule transforms continuous version of PSO to discrete version. Proposed methodology is named as DSAPSO. To check the efficiency of proposed methodology the results of DSAPSO is compared with the results of genetic algorithm (GA).

General Terms
Nature Inspired Algorithms, optimization problem, Computational Biology.

Keywords
DNA sequence assembly, Particle Swarm Optimization, PSO, Swarm Intelligence, SPV, Bioinformatics.

1. INTRODUCTION
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.

The one of the most common problem in biology is DNA sequence assembly problem. In DNA sequence assembly problem, DNA sequence is breaks into number of fragments and after removing duplicate sequences obtain a common consensus sequence. DNA sequence assembly problem is very complex problem and take more computational time to obtain consensus sequence. DNA sequence assembly problem is NP hard problem because there are lots of solutions available for this kind of problem. Many literatures provide solutions for DNA sequence assembly problem. Christian Burks [4] is widely held that DNA sequencing throughput will have to be increased by orders of magnitude to complete the task in the time frame of 15 years that was laid out for the Human Genome Project, and that such dramatic increases will rely in large part on automating the several experimental and interpretive steps involved in DNA sequencing. Over the past decade a number of fragment assembly packages have been developed and used to sequence different organisms. The most popular packages are PHRAP [5] is a program for assembling shotgun DNA sequence data. TIGR assembler [6] overcomes several major obstacles to assembling DNA sequence. STROLL [7] implemented a reliable technique to sequence DNA using primer walking approach. CAP3 [8] includes a number of improvements and new features to improve DNA sequence assembly. Celera assembler [9] developed at Celera for the 2001 publication of the first draft human genome sequence. EULER [10] is an approach to fragment assembly that abandons the classical "overlap - layout - consensus" paradigm that is used in all currently available assembly tools. Allex, C. F., Baldwin, S**.** F., Sbavlik, J. W. and Blamer, F. R. [11] is improving the quality of automatic DNA sequence assembly using fluorescent trace-data classifications. Wilks, C.and Khuri, S. [12] proposed "A Structured Pattern Matching Approach to Shotgun Sequence Assembly," (AMASS) created by Sun Kim.

Many heuristic approaches are applied in DNA Sequence Assembly which are improved the process of DNA Sequence Assembly one of them is Genetic Algorithm. Parsons, R. and Forrest, S. and Burks, C. [13] presents that the genetic algorithm is a promising method for fragment assembly problems, achieving usable solutions quickly. Parsons, R.J. and Forrest, S. and Burks, C. [14] study different genetic algorithm operators for one permutation problem associated with the Human Genome Project—the assembly of DNA sequence fragments from a parent clone whose sequence is unknown into a consensus sequence corresponding to the parent sequence. Parsons, R.J. and Johnson, M.E. [15] discussed the new results, the modifications to the previous genetic algorithm used, the experimental design process by which the new results were obtained, the questions raised by these results, and some preliminary attempts to explain these results. Kim, K. and Mohan, CK [16] presents a fragment assembler using a new parallel hierarchical adaptive variation of evolutionary algorithms. The innovative features include a new measure for evaluating sequence assembly quality and the development of a hybrid algorithm. Fang, S.C. and Wang, Y. and Zhong, J. [17] approach maximizes the similarity (overlaps) between given fragments and a candidate sequence. It considers both whole fragments and the single basepair similarities in the sequence. Special genetic operators are designed to speed up the searching process. Kikuchi, S.and Chakraborty, G. [18] added two

heuristic ideas with GA to make it more efficient. One is chromosome reduction (CRed) step which shorten the length of the chromosomes, participating in genetic search, to improve the efficiency. The other is chromosome refinement (CRef) step which is a greedy heuristics, rearranging the bits using domain knowledge, to locally improve the fitness of chromosomes. Luque, G. and Alba, E. [19] present several methods, a canonical genetic algorithm, a CHC method, a scatter search algorithm, and a simulated annealing, to solve accurately problem instances that are 77K base pairs long. Meksangsouy, P. and Chaiyaratana, N. [20] proposed an asymmetric ordering representation where a path co-operatively generated by all ants in the colony represents the search solution. Zhao, Y. and Ma, P. and Lan, J. and Liang, C. and Ji, G. [21] improved sequence alignment method based on the ant colony algorithm. The new method could avoid a local optimum and remove especially the paths scores of great difference by regulating the initial and final positions of ants and by modifying pheromones in different times.

There are few literatures available which represent solution for DNA Sequence Assembly problem using metaheuristic and nature inspired algorithms. PSO algorithm comes under nature inspired algorithm and it is very effective technique to solve an optimization problem. It has been proven that PSO solves Optimization problem efficiently and gives the optimum result. PSO algorithm used to solve computational biology problem and gives better result than the conventional methods. DNA sequence assembly problem is not solved previously by PSO algorithm that's why we have intention to apply PSO algorithm for DNA Sequence Assembly problem.

2. DNA SEQUENCE ASSEMBLY PROBLEM

Deoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms and some viruses. The main role of DNA molecules in living organism is the long-term storage of information.

DNA sequence is represented by a string of characters drawn from a four-letter alphabet **(A, C, G,** and **T**) corresponding to the four monomeric bases of which the DNA polymer is composed. A piece, or fragment, corresponds in our context to a substring of 100-1000 bases. Overlap strength and offset relationships between pairs of fragments, used to drive the assembly of the fragments into a global layout, is based on comparison of character strings. The output generated by sequencing represents a consensus on the order of 1000-1,000,000 bases long, generated by voting in aligned columns of bases resulting from the layout.

The assembly problem is a combinatorial optimization problem where the aim of the search is to find the right order and orientation of each fragment in the fragment ordering sequence that leads to the formation of a consensus sequence.

Figure 1, shows the basic DNA sequence assembly process. Fig 1(A) represents 6 different DNA fragment taken from large human DNA sequence STIM1. STIM1 DNA sequence is taken from NCBI. After getting 6 fragment from large DNA sequence file arrangement of fragments is needed to calculate consensus sequence.

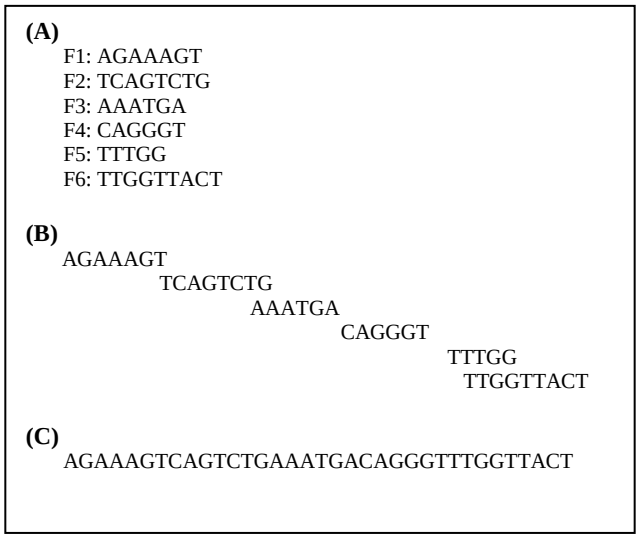

Fig 1: (A) Shows the 6 fragments taken from the file STIM1 DNA sequence (B) Show the fragment order to calculate consensus fragment sequence (C) Common Consensus sequence calculated.

Fig 1(B) represents the order of fragments, fragments are arranged in a manner that suffix of one fragment is compared with prefix of another fragment. Matched nucleotides are removed and remaining nucleotides are considered as consensus sequence. Fig 1(C) represents the calculated consensus sequence.

3. 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. 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. 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 $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})^T$. 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. 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 [2].

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.

4. METHODOLOGY

In this paper we have proposed a solution for DNA sequence assembly problem using particle swarm optimization. For solving any optimization problem we have to first formulate the problem according to optimization problem. In this case first we formulate the DNA sequence assembly problem according to PSO algorithm. Next subsection describes how we formulate the DNA sequence assembly problem.

To solve the problem, representation of the individual and fitness value is required. PSO algorithm is based on population (candidate solution) and each population have its own fitness value according to which it is compared from others, so we have to first represent the DNA sequence assembly problem in terms of PSO algorithm.

4.1 Individual Representation

In DNA sequence assembly problem inputs are the set of fragments which need to be assembled and build a common sequence which does not have any repeated pattern. The common sequence is considered as output for DNA sequence assembly problem. To find out the function or property of specific genes, the reading of nucleotide or chemical base (A, T, C, G) sequence is done. Large nucleotide sequences are called DNA sequence. The large DNA sequence consists of repeated patterns of nucleotide that's why the DNA sequence becomes large. In DNA sequence assembly repeated patterns are removed and one consensus sequence builds.

In DNA sequence assembly large DNA sequence of a particular gene is taken for assembly process. Large DNA file is split randomly in different fragments of DNA sequence which are used in assembly process. After getting the set of fragment, fragments are aligned and the longest match between the suffix of one sequence and the prefix of another is determine. All possible pair combination of fragments is compared and matching score is determined. On the basis of matching score fragment order is determined. At last the consensus sequence is found out from the fragment order. We have performed experiments with the nucleotide sequences of homosapiens(human) and mouse viz. MACF1, TNFRSF19 and Zfa. The DNA data is taken from the NCBI.

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 $X_{id} = \{ x_1, x_2, x_3, \ldots\ldots x_d \}$ 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 sequence vector using SPV is represented as $S_{id} = [f_{i1}, f_{i2}, \ldots\ldots f_{id}]$. 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.

4.2 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. (4).

$$score_{i,i+1} = \begin{cases} 0, & if\ nucleotide\ does\ not\ matched \\ score_{i,i+1} + 1, & otherwise \end{cases} \quad (4)$$

In eq. (4) $score_{i,i+1}$ 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. (5).

$$max\ f_i(x) = \sum_{j=0}^{D-1} score_{j,j+1} \quad (5)$$

In eq. (5) $f_i(x)$ 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. (4) for an individual.

4.3 DSAPSO: DNA Sequence Assembly using PSO Algorithm

We have used particle swarm optimization algorithm 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 represents a solution and has a dimension value. For DNA sequence assembly dimension of PSO individual is equal to the number of fragments taken.

PSO 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.

5. EXPERIMENT RESULT & 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.

5.1 Experimental Setup

To work algorithm in efficient manner first the parameter related with algorithms need to be set. DSAPSO algorithm has a few control parameters: Maximum number of function evaluation (MaxEval), we have tested our algorithms for three different function evaluation value 3000, 5000 and 10000. For experiment we have taken the swarm size or number of individual is equal to 30. In DNA sequence assembly problem individual dimension is equal to the number of fragments. We have tested our algorithms for different number of fragment i.e. dimensions value 10, 15, and 30 and. Size of a fragment is different for different fragment from 50 to 100. Another control parameter is number of runs and we have taken its value in our experiment as 30. It must be noted that each run contains maximum function evaluation. The next control parameter is the value of c1 & c2 which we have taken as 1.14. And w (Inertia weight) is also a control parameter and we have taken its value as 0.7. DNA sequence we used for experiment is MACF1, TNFRSF19 and Zfa. We calculated the consensus sequence for the number of fragments 10, 20 and 30. The best function values of the best solutions found in 30 runs by the algorithm for different dimensions have been recorded.

5.2 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

5.3 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 DSAPSO algorithm we have compared the results of DSAPSO with genetic algorithm.

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 DSAPSO and GA. 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.

TABLE 2. Matching Score comparison using dataset MACF1

		Number of Fragments					
		10		15		30	
DATA SET	**MaxE val**	**DSA PSO**	**GA**	**DSA PSO**	**GA**	**DSA PSO**	**GA**
MACF 1	3000	14	14	20	14	30	28
	5000	16	15	23	16	39	36
	10000	20	16	25	19	40	37

TABLE 3. Matching Score comparison using dataset TNFRSF19

		Number of Fragments					
		10		15		30	
DATA SET	**MaxE val**	**DSA PSO**	**GA**	**DSA PSO**	**GA**	**DSA PSO**	**GA**
TNFRS F19	3000	16	7	96	39	112	94
	5000	55	10	102	41	115	102
	10000	97	30	110	44	125	116

TABLE 4. Matching Score comparison using dataset Zfa

		Number of Fragments					
		10		15		30	
DATA SET	**MaxE val**	**DSA PSO**	**GA**	**DSA PSO**	**GA**	**DSA PSO**	**GA**
Zfa	3000	47	23	75	68	164	116
	5000	52	25	91	72	182	155
	10000	74	25	118	73	227	167

It is clear from the table 2, 3 and 4 that PSO with SPV performs better than the genetic algorithm. PSO with SPV gives the better matching score than the GA. 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.

6. CONCLUTION & FUTURE WORK

It can be concluded from the above results that the DSAPSO 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 PSO 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 PSO is changed to the discrete for changing the real value to discrete SPV rule is used. The fitness value is considered as the addition of the matching score of consecutive fragment from the sequence order generated by the SPV rule using the position value of PSO. PSO updates its solution in each iteration and new real value generated. After modification of PSO 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 genetic algorithm.

In future we have intension to apply various nature inspired algorithms for DNA sequence assembly problem and compare their results with our proposed DSAPSO 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.

7. REFRENCES

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

[2] 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.

[3] 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.

[4] Burks, C. "DNA sequence assembly", Engineering in Medicine and Biology Magazine, IEEE, Vol. 13, pp. 771—773, 1994.

[5] P. Green. Phrap. http://www.phrap.org/.

[6] 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.

[7] T. Chen and S. Skiena. "Trie-based data structures for sequence assembly", Combinatorial Pattern Matching, pp. 206–223, 1998.

[8] X. Huang and A. Madan. "CAP3: A DNA sequence assembly program", Genome Research, Vol. 9, pp. 868–877, 1999.

[9] E.W. Myers. "Towards simplifying and accurately formulating fragment assembly", Journal of Computational Biology, Vol. 2, pp. 275–290, 2000.

[10] P.A. Pevzner. "Computational molecular biology: An algorithmic approach", The MIT Press, Vol. 1, 2000.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

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