2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

A Reinforcement Learning Approach for Solving the Fragment Assembly Problem

Maria-Iuliana Bocicor, Gabriela Czibula, and Istvan-Gergely Czibula
Department of Computer Science, Babeş-Bolyai University
1, M. Kogălniceanu Street
400084, Cluj-Napoca, Romania
Email: {iuliana, gabis, istvanc}@cs.ubbcluj.ro

***Abstract*—The *DNA fragment assembly* is a very complex optimization problem important within many fields including bioinformatics and computational biology. The problem is *NP-hard*, that is why many computational techniques including computational intelligence algorithms were designed for finding good solutions for this problem. Since DNA fragment assembly is a crucial part of any sequencing project, researchers are still focusing on developing better assemblers. In this paper we aim at proposing a new reinforcement learning based model for solving the *fragment assembly* problem. We are particularly focusing on the *DNA fragment assembly* problem. Our model is based on a *Q-learning* agent-based approach. The experimental evaluation confirms a good performance of the proposed model and indicates the potential of our proposal.**

***Keywords*-bioinformatics; reinforcement learning; DNA fragment assembly;**

I. Introduction

The process of DNA sequencing has nowadays become of great importance in basic biology research, as well as in various fields such as medicine, biotechnology or forensic biology. Several techniques have been developed to achieve the DNA sequencing, but the main problem with the current technology is that it cannot read an entire genome at once, not even more than 1000 bases.

The *DNA fragment assembly* is a technique that attempts to reconstruct the original DNA sequence from a large number of fragments, each several hundred base-pairs long [1]. It is an NP-hard combinatorial optimization problem which is growing in importance and complexity as more research centers become involved on sequencing new genomes [2]. Various heuristics, including computational intelligence algorithms, have been designed for solving the fragment assembly problem, but since this problem is a crucial part of any sequencing project, better assemblers are needed [2].

In this paper we propose a reinforcement learning based model for solving the *DNA Fragment Assembly* problem. *Reinforcement Learning* (RL) [3] is an approach to machine intelligence in which an agent can learn to behave in a certain way by receiving punishments or rewards on its chosen actions. To our knowledge, except for the ant [4] based approaches, the *DNA Fragment Assembly* problem has not been addressed in the literature using reinforcement learning, so far. Even if we are focusing on the *DNA Fragment Assembly* problem, the model proposed in this paper can be simply applied to a general *fragment assembly* problem.

The rest of the paper is organized as follows. Section II presents the main aspects related to the topics approached in this paper, the *DNA fragment assembly* problem and *reinforcement learning*. The reinforcement learning model that we propose for solving the DNA fragment assembly problem is introduced in Section III. An experiment is given in Section IV and in Section V we provide an analysis of the proposed reinforcement model, emphasizing its advantages and drawbacks. Section VI outlines some conclusions of the paper and indicates future research directions.

II. Background

In this section we briefly review the fundamentals of the *DNA fragment assembly* problem and *reinforcement learning*.

A. Reinforcement learning

The goal of building systems that can adapt to their environments and learn from their experiences has attracted researchers from many fields including computer science, mathematics, cognitive sciences [3]. *Reinforcement Learning* (RL) [5] is an approach to machine intelligence that combines two disciplines to solve successfully problems that neither discipline can address individually: *Dynamic programming* and *Supervised learning*.

RL is a synonym of learning by interaction [6]. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them. During learning, the adaptive system tries some actions on its environment, then, it is reinforced by receiving a scalar evaluation (the *reward*or *reinforcement* of its actions; the reward is received at the end, in a terminal state, or in any other state, where the agent has correct information about what he did well or wrong. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time. Reinforcement learning tasks are generally treated in discrete time steps. In RL, the computer is simply given a goal to achieve. The computer then learns how to achieve that goal by trial-and-error interactions with its environment.

978-0-7695-4630-8/11 $26.00 © 2011 IEEE
DOI 10.1109/SYNASC.2011.9

The agent's goal, in a RL task is to maximize the sum of the reinforcements received when starting from some initial state and proceeding to a terminal state.

A reinforcement learning problem has three fundamental parts [3]:

- *The environment* - represented by "states". Every RL system learns a mapping from situations to actions by trial-and-error interactions with the environment.
- *The reinforcement function* - the "goal" of the RL system is defined using the concept of a reinforcement function, which is the exact function of future reinforcements the agent seeks to maximize. In other words, there exists a mapping from state/action pairs to reinforcements. After performing an action in a given state, the RL agent will receive some reinforcement (reward) in the form of a scalar value. The RL agent learns to perform actions that will maximize the sum of the reinforcements received when starting from some initial state and proceeding to a terminal state.
- *The value (utility) function* - explains how the agent learns to choose "good" actions, or even how we might measure the utility of an action. Two terms are defined: a policy determines which action should be performed in each state. The value of a state is defined as the sum of the reinforcements received when starting in that state and following some fixed policy to a terminal state. The value (utility) function would therefore be the mapping from states to actions that maximizes the sum of the reinforcements when starting in an arbitrary state and performing actions until a terminal state is reached.

At each time step, t, the learning system receives some representation of the environment's state s, it takes an action a, and one step later it receives a scalar reward r_t, and finds itself in a new state s'. The two basic concepts behind reinforcement learning are trial and error search and delayed reward [7]. The agent's task is to learn a control policy, $\pi : S \rightarrow A$, that maximizes the expected sum E of the received rewards, with future rewards discounted exponentially by their delay, where E is defined as $r_0 + \gamma \cdot r_1 + \gamma^2 \cdot r_2 + ...$ ($0 \leq \gamma < 1$ is the discount factor for the future rewards).

One key aspect of reinforcement learning is a trade-off between *exploitation* and *exploration* [8]. To accumulate a lot of reward, the learning system must prefer the best experienced actions, however, it has to try (to experience) new actions in order to discover better action selections for the future. There are two basic RL designs to consider:

- The agent learns a *utility function* (*U*) on states (or states histories) and uses it to select actions that maximize the expected utility of their outcomes.
- The agent learns an *action-value function* (*Q*) giving the expected utility of taking a given action in a given state. This is called *Q-learning*.

B. The Fragment Assembly Problem

The *Fragment Assembly* (FA) problem deals with sequencing of DNA [2]. In order to sequence larger strands of DNA, they are first broken into smaller pieces. The FA problem is then to reconstruct the original molecules sequence from the smaller fragment sequences [2].

1) Problem Definition and Relevance: The genome of all living organisms, encoded in the DNA, represents the totality of their hereditary information. DNA, or the deoxyribonucleic acid is composed of complex organic molecules and is the information-bearing molecule in the cell - it stores information that contains the instructions needed to construct other components of the cell such as proteins and RNA, and eventually, entire organisms.

Determining the order of nucleotide bases, or the process of DNA sequencing, has nowadays become of great importance in basic biology research, as well as in various fields such as medicine, biotechnology or forensic biology. Several techniques have been developed to achieve the DNA sequencing, but the main problem with the current technology is that it cannot read an entire genome at once, not even more than 1000 bases. As even the simplest organisms (such as viruses or bacteria) have much longer genomes, the need to develop methods that would overcome this limitation arose. One of these, called *shotgun sequencing* was introduced in 1982, by Fred Sanger [9] and it consists of the next steps: first, several copies of the DNA molecule are created; then each of the copies is cut at random sites in order to obtain molecules short enough to be sequenced directly - fragments; the last and most difficult step involves assembling these molecules back into the original DNA molecule, based on common subsequences of fragments.

The DNA Fragment Assembly Problem specifically refers to this last step. Usually, there are three phases that must be followed in order to obtain a solution for this problem [1]:

- *Overlap Phase* - consists of finding the longest match between the prefix of a fragment and the suffix of another. All possible pairs of fragments are compared, in order to determine their similarity.
- *Layout Phase* - refers to finding the order of fragments. This is the most difficult step, because of various challenges:
  1) **Unknown orientation**: the fragments' orientation is lost, so if for a specific fragment there are no overlapping fragments, there might be some that are similar with the fragment's reverse.
  2) **Base call errors**: these are errors that might appear during the experimental phase. There are three types of such errors: substitutions (a certain nucleotide base is subtituted by another), insertions (a new nucleotide base is inserted to a fragment it does not belong to) and deletions (certain nucleotide bases are deleted from frag-

ments). These errors affect the overlaps between the fragments.

3) **Incomplete coverage**: The given set of fragments cannot form the orginial DNA.
4) **Repeated regions**: some subsequences may be repeated several times in the original DNA. These repeats can cause serious problems in the assembly process.
5) **Chimeras**: these occur when two fragments that are not adjacent in the original molecule join together in one fragment.

- *Consensus Phase* - consists of determining the original DNA molecule from the layout obtained in the previous phase.

Next, we will illustrate the assembly process, by using a simple example, taken from [10]. Let us assume that, for the DNA sequence $TTACCGTGC$ we are given the set of fragments:

$F_1 = ACCGT$ $\quad$ $F_3 = TTAC$
$F_2 = CGTGC$ $\quad$ $F_4 = TACCGT$

First we need to determine the overlap of each fragment with the other three fragments. This is usually done using an alignment algorithm and a similarity measure. Then, we need to find the order of these fragments, based on the computed similarities. The order is: $F_3F_4F_1F_2$.

F_3	$\rightarrow$	T	T	A	C	$-$	$-$	$-$	$-$	$-$
F_4	$\rightarrow$	$-$	T	A	C	C	G	T	$-$	$-$
F_1	$\rightarrow$	$-$	$-$	A	C	C	G	T	$-$	$-$
F_2	$\rightarrow$	$-$	$-$	$-$	$-$	C	G	T	G	C
		T	T	A	C	C	G	T	G	C

2) Related Work: Many algorithms have already been developed in the literature to solve the *DNA fragment assembly* problem. The FA problem is known to be NP-complete [11], therefore exact solutions for this problem are very difficult to obtain. For this reason, various heuristic methods have been applied: genetic algorithms, clustering algorithms, ant colony optimization algorithms.

Kikuchi and Chakraborty approach the Fragment Assembly Problem using an improved genetic algorithm [12]. A chromosome is represented as a permutation of the DNA fragments' labels (numbers from 1 to N, where N is the total number of fragments), while the fitness function is represented as the sum of similaritiy measures of all adjacent fragments. The authors add two new methods to their genetic algorithm, in order to improve its efficiency: The Chromosome Reduction Step - used to make the search more efficient (the fragments which are contained within contigs formed in the best chromosome are deleted from all the other chromosomes, thus decreasing their lengths); The Chromosome Refinement Step - greedy heuristic to locally improve the fitness of chromosomes (if the beginning of fragment i overlaps with the end of fragment $i+1$ in the chromosome, then these fragments are swapped).

A four-phase approach is proposed in [13] in order to reconstruct a DNA sequence from fragments. The authors firstly construct an overlap graph, whose vertices contain the fragments and whose edges contain the approximate overlaps between every pair of fragments. Secondly, they try to find an oriented subgraph, by assigning an orientation to each fragment, in order to eliminate some overlaps and retain the possibility of using others. As this phase is NP-complete, the authors proposed a new greedy approximation algorithm that computes an optimal orientation. The third phase consists of selecting a set of edges from an oriented subgraph determined in the previous phase to induce a consistent layout of the oriented fragments. The final step is to determine the original DNA sequence by merging the selected overlaps into a multiple sequence alignment and voting on a consensus.

Parsons, Forrest and Burks [14] argue that the FA problem is quite similar to the Travelling Salesman Problem (TSP) and that genetic algorithms could provide an efficient approach. A solution is represented as a permutation of integer numbers, which represent fragments, and where each two successive fragments overlap. The authors use two fitness functions, one that simply sums the overlap scores over all adjacent fragments - which has to be maximized and another one, that considers the overlap strength among all possible pairs, penalizing layouts in which fragments with strong overlap are far apart - which has to be minimized. New genetic operators that try to preserve or extend existing contigs are defined: order crossover, edge recombination, inversion and transposition.

An ant colony system algorithm was used by Meksangsouy and Chaiyaratana [15] in order to solve the FA problem. The solution is cooperatively generated by all ants in the colony and the performance measure is determined as the sum of the overlap scores calculated for each successive pair of adjacent fragments in the final layout. The authors investigated two types of assembly problems: single-contig and multiple-contig problems, obtaining very good results, especially for the latter.

Angeleri et al. [16] presents a supervised learning approach to the fragment assembly problem. The idea is to train a recurrent neural network to track a sequence of bases constituting a given fragment and to assign to the same cluster all sequences which are well tracked by this network.

III. A Reinforcement Learning Model for Solving the DNA Fragment Assembly Problem

In this section we introduce our reinforcement learning model proposal for solving the *DNA Fragment Assembly* problem.

A general RL task is characterized by four components:

1) a *state space* $\mathcal{S}$ that specifies all possible configurations of the system;
2) the *action space* $\mathcal{A}$ that lists all available actions for the learning agent to perform;
3) the *transition function* δ that specifies the possibly stochastic outcomes of taking each action in any state;
4) a *reward function* that defines the possible reward of taking each of the actions.

A. *The proposed* RL *task for the* FA *problem*

Let us consider, in the following, that $\mathcal{S}eq$ is a DNA sequence and $F_1, F_2, \ldots, F_n$ is a set of fragments. As indicated in Subsection II-B1, the FA problem consists of determining the order in which these fragments have to be assembled back into the original DNA molecule, based on common subsequences of fragments. Consequently, the FA problem can be viewed as the problem of generating a permutation σ of $\{1, 2, \ldots, n\}$ that optimizes the performance of the alignment $F_\sigma = (F_{\sigma_1}, F_{\sigma_2}, \ldots, F_{\sigma_n})$ $(n > 1)$. The performance measure PM we consider in this paper is one of the fitness functions defined in [14], which sums the overlap scores over all adjacent fragments and which has to be maximized.

According to [14], the performance measure PM for the sequence of fragments $F_\sigma = (F_{\sigma_1}, F_{\sigma_2}, \ldots, F_{\sigma_n})$ is defined as in Equation (1):

$$PM(F_\sigma) = \sum_{i=1}^{n-1} w(F_{\sigma_i}, F_{\sigma_{i+1}}) \tag{1}$$

where $w(a, b)$ denotes the similarity measure between sequences a and b.

We define the RL task associated to the *FA* problem as follows:

- The state space $\mathcal{S}$ (the agent's environment) will consist of $\frac{n^{n+1}-1}{n-1}$ states, i.e $\mathcal{S} = \{s_1, s_2, ..., s_{\frac{n^{n+1}-1}{n-1}}\}$. The *initial state* of the agent in the environment is s_1. A state $s_{i_k} \in \mathcal{S}(i_k \in [1, \frac{n^{n+1}-1}{n-1}])$ reached by the agent at a given moment after it has visited states $s_1, s_{i_1}, s_{i_2}, ...s_{i_n}$ and has selected actions $a_{i_1}, a_{i_2}, ...a_{i_n}$ is a *terminal* (final or goal) state if the number of states visited by the agent in the current sequence is $n + 1$ and all the actions are distinct, i.e $a_{i_j} \neq a_{i_k},\ \forall 1 \leq j, k \leq n,\ j \neq k$.
- The action space $\mathcal{A}$ consists of n actions available to the problem solving agent and corresponding to the n possible values $1, 2, \ldots, n$ used to represent a solution (permutation of {1,2,...,n}), i.e $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, where $a_i = i,\ \forall 1 \leq i \leq n$.
- The transition function $\delta : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ between the states is defined as in Formula (2).

$$\delta(s_j) = \bigcup_{k=1}^{n} \Delta(s_j, a_k)\ \forall j, 1 \leq j \leq \frac{n^n - 1}{n - 1}, \tag{2}$$

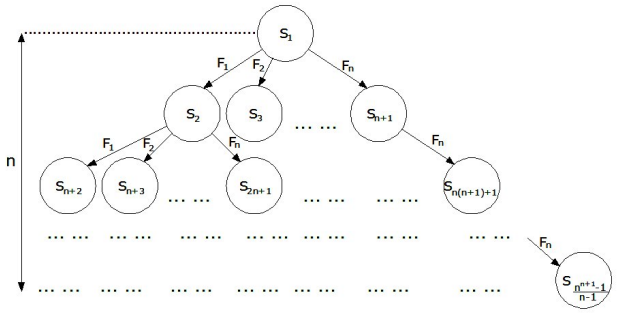

Figure 1. The states space.

where $\Delta(s_j, a_k) = s_{n \cdot j - n + 1 + k}\ \forall k \in [1, n]$.
This means that, at a given moment, from a state $s \in \mathcal{S}$ the agent can move in n successor states, by executing one of the n possible actions. We say that a state $s' \in \mathcal{S}$ that is accessible from state s, i.e $s' \in \bigcup_{a \in \mathcal{A}} \Delta(s, a)$, is the *neighbor* (*successor*) state of s.
The transitions between the states are equiprobable, the transition probability $P(s, s')$ between a state s and each neighbor state s' of s is equal to $1/n$, as each state from $\mathcal{S}$ has n possible successor states (see Formula (2)).
- The reward function will be defined below (Formula (3)).

A graphical representation of the states space for the FA problem associated to the problem of assembling the fragments $F_1, F_2, \ldots, F_n$ into the DNA sequence $\mathcal{S}eq$ is given in Figure 1. The circles represent states and the transitions between states are indicated by arrows labeled with the action that leads the agent from a state to another.
[!t]

Let us consider a path π in the above defined evironment from the initial to a final state, $\pi = (\pi_0 \pi_1 \pi_2 \cdots \pi_n)$, where $\pi_0 = s_1$ and $\forall 0 \leq k \leq n - 1$ the state π_{k+1} is a *neighbor* of state π_k. The sequence of actions obtained following the transitions between the successive states for path π will be denoted by $a_\pi = (a_{\pi_0} a_{\pi_1} a_{\pi_2} \cdots a_{\pi_{n-1}})$, where $\pi_{k+1} = \Delta(\pi_k, a_{\pi_k}),\ \forall 0 \leq k \leq n - 1$. The sequence a_π will be referred as the *action configuration* associated to the path π. A path π is called *valid* if all the actions within its *action configuration* are distinct, i.e $a_{\pi_j} \neq a_{\pi_k},\ \forall 0 \leq j, k \leq n-1,\ j \neq k$. The *action configuration* a_π associated to a *valid* path π can be viewed as a possible order for the fragments assembly process, i.e a permutation that gives the assembly order (the fragments are assembled in the order $F_{a_{\pi_0}}, F_{a_{\pi_1}}, \ldots, F_{a_{\pi_{n-1}}}$). Consequently we can associate to a *valid* path π a value denoted by $PM(F_{a_\pi})$ representing the performance measure (see Equation (1)) of the alignment $F_{a_\pi} = (F_{a_{\pi_0}}, F_{a_{\pi_1}}, \ldots, F_{a_{\pi_{n-1}}})$.

The *FA* problem formulated as a RL problem will consist

of training the agent to find a path π from the initial to a final state having the maximum associated performance measure $PM(F_{a_\pi})$. It is known that the estimated utility of a state [17] in a reinforcement learning process is the estimated *reward-to-go* of the state (the sum of rewards received from the given state to a final state). So, after a reinforcement learning process, the agent learns to execute those transitions that maximize the sum of rewards received on a path from the initial to a final state.

As we aim at obtaining a *valid* path π having the maximum associated performance measure, we define the reinforcement function as follows (Formula (3)):

- the reward received after a transition to a non terminal state is τ, where τ is a small positive constant (e.g. 0.1);
- the reward received after a transition to a final state π_n after states $\pi_0 = s_1, \pi_1, \pi_2, ...\pi_{n-1}$ were visited is the performance measure PM of the alignment $F_{a_\pi} = (F_{a_{\pi_0}}, F_{a_{\pi_1}}, \ldots, F_{a_{\pi_{n-1}}})$ (see Equation (1)).

$$r(\pi_k|s_1, \pi_1, \pi_2, ...\pi_{k-1}) = \begin{cases} PM(F_{a_\pi}) & \text{if } k = n \\ \tau & otherwise \end{cases}, \quad (3)$$

where by $r(\pi_k|s_1, \pi_1, \pi_2, ...\pi_{k-1})$ we denote the reward received by the agent in state π_k, after its history in the environment is $\pi_0 = s_1, \pi_1, \pi_2, ...\pi_{k-1}$.

Considering the reward defined in Formula (3), as the learning goal is to maximize the total amount of rewards received on a path from the initial to a final state, it can be easily shown that the agent is trained to find a path π that maximizes the performance of the associated alignment.

B. The learning process

During the training step of the learning process, the agent will determine its *optimal policy* in the environment, i.e the *policy* that maximizes the sum of the received rewards.

For training the FA (*Fragment Assembly*) agent, we propose a Q-learning approach [3]. The idea of the training process is the following:

- The Q values are initialized with 0.
- During some training episodes, the agent will experiment (using the ϵ-Greedy action selection mechanism) some (possible optimal) *valid* paths from the initial to a final state, updating the Q-values estimations according to the $Q - learning$ algorithm [18].
- At the end of the training process, the Q-values estimations will be in the vicinity of the exact values.

After the training step of the agent has been completed, the solution learned by the agent is constructed by starting from the initial state and following the $Greedy$ mechanism until a solution is reached. From a given state i, using the $Greedy$ policy, the agent transitions to a neighbor j of i having the maximum Q-value. Consequently, the solution of the *FA* problem reported by the RL agent is a path $\pi = (s_1\pi_1\pi_2\cdots\pi_n)$ from the initial to a final state, obtained following the policy described above. We mention that there may be more than one optimal policy in the environment determined following the $Greedy$ mechanism described above. In this case, the agent may report a single optimal policy or all optimal policies, according to the way it was designed.

It is proven in [19] that the learned Q-values converge to their optimal values as long as all state-action pairs are visited an infinite number of times. Consequently, the action configuration a_π corresponding to the path π learned by the FA agent converges, in the limit, to the optimal order in which the fragments have to be assembled, indicating the alignment $F_{a_{\pi_0}}, F_{a_{\pi_1}}, \ldots, F_{a_{\pi_{n-1}}}$ having the maximum associated performance measure.

IV. Experimental evaluation

In this section we aim at experimentally evaluating our RL-based approach for solving the *DNA fragment assembly* problem.

A. Example

First, we provide the reader with an easy to follow example illustrating how our approach works. The example is taken from [10] and was described in Subsection II-B1.

In the following, we will present all possible permutations of fragments from the example given in Subsection II-B1, and their associated performance measures, in order to show that our new approach obtains the optimal results. Table I and Table II illustrate, respectively, the overlap (similarity) scores and the sub-sequence alignments for all possible pairs of fragments that can be obtained. These similarity measures and local alignments between two fragments are obtained using the Smith-Waterman algorithm that detects a local alignment by dynamic programming (as parameters for the scoring matrix of the Smith-Waterman algorithm [20], we used: *match*=1, *mismatch*=-0.33 and *gap*=1.33).

	F_1	F_2	F_3	F_4
F_1	-	3.00	2.00	5.00
F_2	3.00	-	1.67	3.00
F_3	2.00	1.67	-	3.00
F_4	5.00	3.00	3.00	-

Table I
THE SIMILARITY SCORES FOR THE FRAGMENTS, OBTAINED BY THE SMITH-WATERMAN ALGORITHM [20].

Table III presents all the possible permutations of the 4 given fragments and the performance measure PM (Equation (1)) for each permutation. It can be seen that the maximum values for the performance measure PM are obtained in two cases: for the alignments $F_2F_1F_4F_3$ and $F_3F_4F_1F_2$. The original DNA is the one indicated by the alignment $F_2F_1F_4F_3$.

No.	Alignment	Similarity score	Similarity score	Similarity score	Scores' sum
1	$F_1F_2F_3F_4$	$w(F_1,F_2)=3.00$	$w(F_2,F_3)=1.67$	$w(F_3,F_4)=3.00$	7.67
2	$F_1F_2F_4F_3$	$w(F_1,F_2)=3.00$	$w(F_2,F_4)=3.00$	$w(F_4,F_3)=3.00$	9.00
3	$F_1F_3F_2F_4$	$w(F_1,F_3)=2.00$	$w(F_3,F_2)=1.67$	$w(F_2,F_4)=3.00$	6.67
4	$F_1F_3F_4F_2$	$w(F_1,F_3)=2.00$	$w(F_3,F_4)=3.00$	$w(F_4,F_2)=3.00$	8.00
5	$F_1F_4F_2F_3$	$w(F_1,F_4)=5.00$	$w(F_4,F_2)=3.00$	$w(F_2,F_3)=1.67$	9.67
6	$F_1F_4F_3F_2$	$w(F_1,F_4)=5.00$	$w(F_4,F_3)=3.00$	$w(F_3,F_2)=1.67$	9.67
7	$F_2F_1F_3F_4$	$w(F_2,F_1)=3.00$	$w(F_1,F_3)=2.00$	$w(F_3,F_4)=3.00$	8.00
8	$F_2F_1F_4F_3$	$w(F_2,F_1)=3.00$	$w(F_1,F_4)=5.00$	$w(F_4,F_3)=3.00$	11.00
9	$F_2F_3F_1F_4$	$w(F_2,F_3)=1.67$	$w(F_3,F_1)=2.00$	$w(F_1,F_4)=5.00$	8.67
10	$F_2F_3F_4F_1$	$w(F_2,F_3)=1.67$	$w(F_3,F_4)=3.00$	$w(F_4,F_1)=5.00$	9.67
11	$F_2F_4F_1F_3$	$w(F_2,F_4)=3.00$	$w(F_4,F_1)=5.00$	$w(F_1,F_3)=2.00$	10.00
12	$F_2F_4F_3F_1$	$w(F_2,F_4)=3.00$	$w(F_4,F_3)=3.00$	$w(F_3,F_1)=2.00$	8.00
13	$F_3F_1F_2F_4$	$w(F_3,F_1)=2.00$	$w(F_1,F_2)=2.00$	$w(F_2,F_4)=3.00$	8.00
14	$F_3F_1F_4F_2$	$w(F_3,F_1)=2.00$	$w(F_1,F_4)=5.00$	$w(F_4,F_2)=3.00$	10.00
15	$F_3F_2F_1F_4$	$w(F_3,F_2)=1.67$	$w(F_2,F_1)=2.00$	$w(F_1,F_4)=5.00$	8.67
16	$F_3F_2F_4F_1$	$w(F_3,F_2)=1.67$	$w(F_2,F_4)=3.00$	$w(F_4,F_1)=5.00$	8.67
17	$F_3F_4F_1F_2$	$w(F_3,F_4)=3.00$	$w(F_4,F_1)=5.00$	$w(F_1,F_2)=3.00$	11.00
18	$F_3F_4F_2F_1$	$w(F_3,F_4)=3.00$	$w(F_4,F_2)=3.00$	$w(F_2,F_1)=3.00$	9.00
19	$F_4F_1F_2F_3$	$w(F_4,F_1)=5.00$	$w(F_1,F_2)=3.00$	$w(F_2,F_3)=1.67$	9.67
20	$F_4F_1F_3F_2$	$w(F_4,F_1)=5.00$	$w(F_1,F_3)=2.00$	$w(F_3,F_2)=1.67$	8.67
21	$F_4F_2F_1F_3$	$w(F_4,F_2)=3.00$	$w(F_2,F_1)=3.00$	$w(F_1,F_3)=2.00$	8.00
22	$F_4F_2F_3F_1$	$w(F_4,F_2)=3.00$	$w(F_2,F_3)=1.67$	$w(F_3,F_1)=2.00$	6.67
23	$F_4F_3F_1F_2$	$w(F_4,F_3)=3.00$	$w(F_3,F_1)=2.00$	$w(F_1,F_2)=3.00$	8.00
24	$F_4F_3F_2F_1$	$w(F_4,F_3)=3.00$	$w(F_3,F_2)=1.67$	$w(F_2,F_1)=3.00$	7.67

Table III
ALL THE POSSIBLE PERMUTATIONS, WITH THEIR ASSOCIATED PERFORMANCE MEASURE PM (EQUATION (1))

	F_1	F_2	F_3	F_4
F_1	-	CGT	AC	$ACCGT$
F_2	CGT	-	$TGC-TAC$	CGT
F_3	AC	$TGC-TAC$	-	TAC
F_4	$ACCGT$	CGT	TAC	-

Table II
THE ALIGNMENTS FOR THE FRAGMENTS, OBTAINED BY THE SMITH-WATERMAN ALGORITHM [20].

1) RL model and results: Let us consider the example mentioned above, the DNA sequence $\mathcal{S}eq = TTACCGTGC$ and four fragments $F_1 = ACCGT, F_2 = CGTGC, F_3 = TTAC, F_4 = TACCGT$, i.e. $n = 4$. We aim at identifying the most appropriate order in which the fragments have to be assembled back into the original sequence $\mathcal{S}eq$. As we have presented in Section III, the states space will consist of 341 states, i.e $\mathcal{S} = \{s_1, s_2, ..., s_{341}\}$.

For applying the RL model introduced in Section III in order to solve the fragment assembly problem formulated above, we have used a software framework that we have previously introduced in [21] for solving combinatorial optimization problems using reinforcement learning techniques.

We have trained the FA agent as indicated in Subsection III-B. We remark the following regarding the parameters setting: the discount factor for the future rewards is $\gamma = 0.9$; the number of training episodes is 240; the ϵ-Greedy action selection mechanism was used. Using the above defined parameters and under the assumptions that the state-action pairs are equally visited during training and that the agent explores its search space (the ϵ parameter is set to 1), two optimal solutions were reported after the training of the FA agent was completed. These solutions were determined starting from state s_1, following the $Greedy$ policy (as we have indicated in Subsection III-A). Both of them are optimal, having the maximum associated performance of 11 (see Table III).

The learned optimal solutions are: (1) the path $\pi = (s_1 s_3 s_{10} s_{41} s_{164})$ having the associated *action configuration* $a_\pi = (2143)$ and (2) the path $\pi = (s_1 s_4 s_{17} s_{66} s_{263})$ having the associated *action configuration* $a_\pi = (3412)$.

B. Experiment

This subsection presents a second example on which we will illustrate our approach. We have chosen a small section of DNA belonging to the bacterium *Escherihia coli (E. coli)*. The DNA sequence contains 25 bases: *TACTAGCAATACGCTTGCGTTCGGT*. Using the Perl scrips that the authors of [22] produced to generate fragments from a given DNA reference, for the above mentioned *E. Coli* sequence, we obtained a number of 10 fragments, each having a length of 8 bases:

$$F_1 = ACGCTTGC \qquad F_6 = TACTAGCA$$
$$F_2 = TTGCGTTC \qquad F_7 = AATACGCT$$
$$F_3 = ACTAGCAA \qquad F_8 = CTTGCGTT$$
$$F_4 = CGTTCGGT \qquad F_9 = ATACGCTT$$
$$F_5 = AGCAATAC \qquad F_{10} = CTAGCAAT$$

These fragments are ordered in the following way to form the original DNA sequence: $F_6F_3F_{10}F_5F_7F_9F_1F_8F_2F_4$. Table IV illustrates the overlap (similarity) scores for all possible pairs of fragments that can be obtained. The similarity measures between the fragments were obtained, as in the previous example, using the same Smith-Waterman algorithm [20] and the same parameters for the scoring matrix: match=1, mismatch=-0.33 and gap=1.33.

The maximum value for the performance measure PM is 55 and is obtained in two cases: for the alignments $F_6F_3F_{10}F_5F_7F_9F_1F_8F_2F_4$ and $F_4F_2F_8F_1F_9F_7F_5F_{10}F_3F_6$. The first alignment indicates the original DNA.

1) RL model and results: For the example presented in this section, we built the states and action spaces of our RL model, in order to be able to assemble the fragments optimally, so as to obtain the original DNA sequence. The states space will consist of $\frac{10^{11}-1}{9}$ states, while the action space will contain 10 actions, corresponding to the 10 given fragments.

We have trained the FA agent as indicated in Subsection III-B. We remark the following regarding the parameters setting: the discount factor for the future rewards is $\gamma = 0.9$; the number of training episodes is $4{\cdot}10^6$; the ϵ-Greedy action selection mechanism was used. Using the above defined parameters and under the assumptions that the state-action pairs are equally visited during training and that the agent explores its search space (the ϵ parameter is set to 1), the solution was reported after the training of the FA agent was completed. This solution is optimal, it has the maximum associated performance of 55 and was determined starting from state s_1, following the $Greedy$ policy (as we have indicated in Subsection III-A).

The learned solution is the alignment that indicates the original DNA, it is the path $\pi = (s_1 s_7 s_{64} s_{641} s_{6406} s_{64058} s_{640580} s_{6405792} s_{64057919} s_{640579183} s_{6405791825})$ having the associated *action configuration* $a_\pi = (6\ 3\ 10\ 5\ 7\ 9\ 1\ 8\ 2\ 4)$.

V. Disscussion

Regarding the Q-learning approach introduced in Section III for solving the DNA fragment assembly problem, we remark the following. The training process during an episode has a time complexity of $\theta(n)$, where n is the number of fragments considered in the assembly process. Consequently, assuming that the number of training episodes is k, the overall complexity of the algorithm for training the FA agent is $\theta(k \cdot n)$. We mention that if the number n of the fragments considered in the assembly problem is large and consequently the state space becomes very large, in order to store the Q values estimates, a neural network should be used.

In the following we will briefly compare our approach with some of the existing approaches. The comparison is made considering the computational time complexity point of view. Since for the most of the existing approaches the authors do not provide the asymptotic analysis of the time complexity of the proposed approaches, we can not provide a detailed comparison.

An improved genetic algorithm is introduced in [12] for solving the fragment assembly problem. An asymptotic analysis of the computational complexity for evolutionary algorithms (EAs) is difficult [23] and is usually done only for particular problems. Anyway, the number of generations (or equivalently the number of fitness evaluations) is the most important factor in determining the order of EA's computation time. In our view, the time complexity of an evolutionary approach for solving the problem of assembling n fragments is at least $noOfRuns \cdot n \cdot noOfGenerations \cdot populationLength$. For large instances, it is likely (even if we can not rigorously prove) that the computational complexity of our approach is less than the one of an evolutionary approach.

Ant Colony Systems were already used for solving the DNA fragment reordering problem by Meksangsouy and Chaiyaratana [15] and Wetcharaporn et al. [24]. Neumann et al. show in [25] how simple ACO algorithms can be analyzed with respect to their computational complexity on example functions with different properties, and also claim that asymptotic analysis for general ACO systems is difficult. In our view, the time complexity of an ACO approach for solving the problem of assembling n fragments is at least $noOfRuns \cdot n \cdot noOfIterations \cdot noOfAnts$. For large instances, it is likely (even if we can not rigurously prove) that our approach has a lower computational complexity.

Compared to the supervised approach proposed by Angeleri et al. [16] and based on training a recurrent neural network, the advantage of our RL model is that the learning process needs no external supervision, as in our approach the solution is learned from the rewards obtained by the agent during its training. It is well known that the main drawback of supervised learning models is that a set of inputs with their target outputs is required, and this can be a problem.

The main drawback of our approach is that a very large number of training episodes has to be considered in order to obtain accurate results and this leads to a slow convergence. In order to speed up the convergence process, further improvements, such as local search mechanisms will be considered. Anyway, we think that the direction of using reinforcement learning techniques in solving the fragment assembly problem is worth being studied and further improvements can lead to valuable results.

VI. Conclusions and Further Work

We have proposed in this paper a reinforcement learning based model for solving the fragment assembly problem. To our knowledge, except for the ant based approaches, the *fragment assembly* problem has not been addressed in

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
F_1	-	4	3.67	3.34	2.68	3.67	5	5	6	3.67
F_2	4	-	2.67	5	2.34	2.67	3.34	7	4.01	3.01
F_3	3.67	2.67	-	1.67	5	7	4.34	3.67	3.34	7
F_4	3.34	5	1.67	-	1.67	1.68	3.68	5.01	3.34	2.35
F_5	2.68	2.34	5	1.67	-	4	5	2.34	4	6
F_6	3.67	2.67	7	1.68	4	-	3.34	3.67	3	6
F_7	5	3.34	4.34	3.68	5	3.34	-	3.68	7	3
F_8	5	5	3.67	5.01	2.34	3.67	3.68	-	3.34	5.01
F_9	6	4.01	3.34	3.34	4	3	7	3.34	-	3.34
F_{10}	3.67	3.01	7	2.35	6	6	3	5.01	3.34	-

Table IV
THE SIMILARITY SCORES FOR THE FRAGMENTS, OBTAINED BY THE SMITH-WATERMAN ALGORITHM [20].

the literature using reinforcement learning, so far. We have emphasized the potential of our proposal, highlighting its advantages and drawbacks.

We plan to extend the evaluation of the proposed RL model for some larger instances, to further test its performance. We will also investigate possible improvements of the RL model by analyzing a temporal difference approach [3], by using different reinforcement functions and by adding different local search mechanisms in order to increase the model's performance. An extension of the FA model to a distributed RL approach will be also considered.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their comments and suggestions to improving the paper. This work was partially supported by CNCSIS - UEFISCDI, project number PNII - IDEI 2286/2008 and the Sectorial Operational Programme for Human Resources Development 2007-2013, co-financed by the European Social Fund, under the project number POSDRU/107/1.5/S/76841 with the title "Modern Doctoral Studies: Internationalization and Interdisciplinarity".

REFERENCES

[1] L. Li and S. Khuri, "A comparison of DNA fragment assembly algorithms," in *Proc. of the Intl Conf. on Mathematics and Engineering Techniques in Medicine and Biological Sciences*. CSREA Press, 2004, pp. 329–335.

[2] A. E. Hassanien, M. G. Milanova, T. G. Smolinski, and A. Abraham, "Computational intelligence in solving bioinformatics problems: Reviews, perspectives, and challenges," in *Computational Intelligence in Biomedicine and Bioinformatics*, 2008, pp. 3–47.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[4] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004.

[5] L. J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, pp. 293–321, 1992.

[6] A. Perez-Uribe, "Introduction to reinforcement learning," 1998, http://lslwww.epfl.ch/∼anperez/RL/RL.html.

[7] D. Chapman and L. P. Kaelbling, "Input generalization in delayed reinforcement learning: an algorithm and performance comparisons," in *Proc. of the 12th International Joint Conference on Artificial intelligence - Volume 2*. Morgan Kaufmann Publishers Inc., 1991, pp. 726–731.

[8] S. Thrun, "The role of exploration in learning control," in *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Florence, Kentucky: Van Nostrand Reinhold, 1992.

[9] F. Sanger, A. Coulson, G. Hong, l. D. Hil, and G. Petersen, "Nucleotide sequence of bacteriophage Lambda DNA," *J. Molecular Biology*, vol. 162, no. 4, pp. 729–773, 1982.

[10] W. Kosters, "Bioinformatics: Fragment assembly," *IPA–Algorithms and Complexity - course*, 2007. [Online]. Available: http://www.liacs.nl/ kosters/bio/bio.pdf

[11] P. A. Pevzner, "Computational molecular biology: An algorithmic approach," 2000.

[12] S. Kikuchi and G. Chakraborty, "Heuristically tuned GA to solve genome fragment assembly problem," *IEEE CEC*, pp. 1491–1498, 2006.

[13] J. D. Kececioglu and E. W. Myers, "Combinatiorial algorithms for DNA sequence assembly," *Algorithmica*, vol. 13, no. 1/2, pp. 7–51, 1995.

[14] R. J. Parsons, S. Forrest, and C. Burks, "Genetic algorithms, operators, and DNA fragment assembly," in *Machine Learning*. Kluwer Academic Publishers, 1995, pp. 11–33.

[15] P. Meksangsouy and N. Chaiyaratana, "DNA fragment assembly using an ant colony system algorithm," in *Proceedings of CEC'03 - vol.3*. IEEE Press, 2003, pp. 1756–1763.

[16] E. Angeleri, B. Apolloni, D. de Falco, and L. Grandi, "DNA fragment assembly using neural prediction techniques," *Int. J. Neural Syst.*, vol. 9, no. 6, pp. 523–544, 1999.

[17] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach*, ser. Prentice Hall International Series in Artificial Intelligence. Prentice Hall, 2003.

[18] P. Dayan and T. Sejnowski, "Td(Lambda) converges with probability 1," *Mach. Learn.*, vol. 14, pp. 295–301, 1994.

[19] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[20] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[21] I. Czibula, M. Bocicor, and G. Czibula, "A software framework for solving combinatorial optimization tasks," *Studia Universitatis "Babes-Bolyai", Informatica, Proc. of KEPT 2011, Special IssueSpecial Issue*, vol. LVI, pp. 3–8, 2011.

[22] W. Zhang, J. Chen, Y. Yang, Y. Tang, J. Shang, B. Shen, and I. K. Jordan, "A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies," *PLoS ONE*, vol. 6, no. 3, p. e17915, 2011.

[23] W. E. Hart and R. K. Belew, "Optimizing an arbitrary function is hard for the genetic algorithm," in *Proc. of the 4th Intern. Conference on Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 190–195.

[24] W. Wetcharaporn, N. Chaiyaratana, and S. Tongsima, "DNA Fragment Assembly: An Ant Colony System Approach," in *Applications of Evolutionary Computing: EvoWorkshops 2006*, ser. LNCS, vol. 3907. Budapest, Hungary: Springer Berlin / Heidelberg, 2006, pp. 231–242.

[25] F. Neumann, D. Sudholt, and C. Witt, "Computational complexity of ant colony optimization and its hybridization with local search," in *Innovations in Swarm Intelligence*, 2009, pp. 91–120.