On the Approximate Period Problem

Anna Gorbenko

***Abstract*—Different regularities can be used to identify the sequence among other sequences. Regularities allow us to infer an information about the evolution of the sequence. Tandem repeats are the most frequent in the genomes of eukaryotes. Extraction of regularities is a widely studied problem. However, searching for exact tandem repeats can be too restrictive. So, a natural extension of the repetition is to allow errors. In this paper, we consider the approximate period problem. In particular, we consider an explicit reduction from the approximate period problem to the satisfiability problem and present experimental results for different satisfiability algorithms. Also, we consider the approximate period problem for sequences of motor primitives of robots. In particular, we use the approximate period problem to obtain some meta-parameters that adapt the global motion behavior. We try to use such meta-parameters for learning to generalize motor primitives to a different behavior by trial and error without re-learning the task.**

***Index Terms*—approximate period, penalty matrix, satisfiability problem, genetic algorithm, robot motor primitives.**

I. Introduction

INVESTIGATION of regularities plays an important role for the detection of different knowledge (see e.g. [1]–[3]). Various methods of extraction of regularities in a biological sequence can be used to identify the sequence among other sequences. Comparison of revealed regularities allows us to infer an information about the evolution of the sequence. The genomes of eukaryotes contain many regularities. Tandem repeats are the most frequent. It should be noted that finding occurrences of repeated substrings in biological sequences is a widely studied problem. In particular, searching for tandem repeats is used to reveal structural and functional information. However, there are a relatively large number of different errors in biological sequences. Therefore, searching for exact tandem repeats can be too restrictive. A natural extension of the repetition is to allow errors.

In this paper, we consider the notion of approximate periods of strings which is an approximate version of periods. This notion is first discussed in [4]. In particular, the approximate period problem was proposed in [4]. In this paper, we consider an approach to solve the problem. In particular, we consider an explicit reduction from the approximate period problem to the satisfiability problem.

II. Preliminaries and the problem definition

The length of a string S is the number of letters in it and is denoted as $|S|$. We use $S[i]$ to denote the ith letter in the string S, and $S[i,j]$ to denote the substring of S consisting of the ith letter through the jth letter. For any alphabet Π, we assume that Π^* is the set of all strings over Π.

Let

$$\Sigma = \{a_1, a_2, \ldots, a_m\}$$

be a fixed alphabet. Throughout the paper, we assume that Γ is augmented alphabet

$$\Gamma = \Sigma \cup \{a_0\}$$

where a_0 is a special symbol. Symbol a_0 is called an indel and represents the insertion or deletion of a particular symbol in one string relative to another.

Traditionally the alignment notation has been used to illustrate a comparison between strings. Let

$$X = \{X_1, X_2, \ldots, X_k\}$$

is a set of strings over Σ. A multiple alignment of X is a set

$$A = \{A_1, A_2, \ldots, A_k\}$$

of strings over Γ such that $|A_i| = n$, A_i is a copy of X_i into which $n - |X_i|$ copies of a_0 have been inserted, for all $1 \leq i \leq k$.

A conventional way to measure the similarity between two strings S and T is to calculate costs of local transformations. Let δ be a distance function. We assume that δ is specified by a penalty matrix. A penalty matrix M specifies the substitution cost for each pair of letters and the insertion and deletion cost for each letter. The weighted edit distance between S and T is the minimum cost to convert S to T using a penalty matrix.

Given two strings X, P and distance function δ, we define approximate periods as follows. If there exists a partition of X into disjoint blocks of substrings, i.e.,

$$X = P_1 \ldots P_r,$$

$P_i \neq \epsilon$, $r > 2$, such that $\delta(P, P_i) < K$ for $1 \leq i < r$, and $\delta(P', P_r) < K$ where P' is some prefix of P, we say that P is a K-approximate period of X (see [4]).

We consider the following problem:

The approximate period problem (AP)

Instance: *A finite alphabet Γ, a string X from Γ^*, a penalty matrix M, and a positive integer K.*

Question: *Is there a string $U \in \Gamma^*$ such that U is a K-approximate period of X?*

Ural Federal University, Department of Intelligent Systems and Robotics of Mathematics and Computer Science Institute, 620083 Ekaterinburg, Russian Federation. Email: gorbenko.aa@gmail.com

The work was partially supported by Analytical Departmental Program "Developing the scientific potential of high school", RFBR, research project No. 13-01-00048 a, and Ural Federal University development program with the financial support of young scientists.

III. An explicit reduction from AP to the satisfiability problem

The problem AP is **NP**-complete [4]. In particular, if $|\Gamma| \geq 9$, then there exists δ such that $\delta(a,a) = 0$, $\delta(a,b) = \delta(b,a)$, for all $a, b \in \Gamma$, and AP is **NP**-complete [4]. It is shown in [5] that AP problem is **NP**-complete for some penalty matrix M and $|\Gamma| \geq 5$. It is shown in [6] that if $|\Gamma| \geq 7$ then there exists δ such that $\delta(a,a) = 0$, $\delta(a,b) = \delta(b,a)$ for all $a, b \in \Gamma$, and AP is **NP**-complete.

The 3-satisfiability problem (3SAT) is the problem of determining if the variables of a given boolean function in conjunctive normal form with 3 variables per clause (3-CNF) can be assigned in such a way as to make the formula evaluate to true (see e.g. [7]). The problem 3SAT is **NP**-complete. However, it should be noted that solving various hard problems with efficient satisfiability algorithms has caused considerable interest (see e.g. [8]–[11]). Since AP is **NP**-complete, it is natural to consider an explicit reduction from AP to the satisfiability problem and try to solve AP with some satisfiability algorithms.

In this paper, we assume that

$$\delta(a,a) = 0,$$

$$\delta(a,b) = \delta(b,a),$$

$$\delta(a,b) + \delta(b,c) \geq \delta(a,c),$$

for all $a, b, c \in \Gamma$. Also, we assume that $\delta(a,b)$ is a nonnegative integer, for all $a, b \in \Gamma$.

Let d denotes

$$\max\{\lceil \log_2 K \rceil, \max_{0 \leq i \leq m, 0 \leq j \leq m} \lceil \log_2 \delta(a_i, a_j) \rceil\}.$$

We assume that

$$w[p] \in \{0,1\},$$

$$w[i,j,p] \in \{0,1\},$$

for all

$$0 \leq i \leq m,$$

$$0 \leq j \leq m,$$

$$0 \leq p \leq d.$$

Also, it is assumed that

$$\delta(a_i, a_j) = \sum_{p=0}^{d} w[i,j,p] 2^p,$$

$$K = \sum_{p=0}^{d} w[p] 2^p.$$

It is clear that if r is a number of disjoint blocks of substrings in a partition of X, then $2 < r \leq |X|$.

Let

$$\bigwedge_{1 \leq i \leq 3} x[1,i], \tag{1}$$

$$\neg x[1, |X| + 1], \tag{2}$$

$$\bigwedge_{1 \leq i \leq |X|} (x[1,i] \vee \neg x[1, i+1]). \tag{3}$$

We use the functions (1) – (3) to select a number r of disjoint blocks of substrings in a partition of X. In particular, we assume that

$$r = i$$

if and only if

$$x[1,i] = 1$$

and

$$x[1, i+1] = 0.$$

Let

$$\bigwedge_{\substack{1 \leq i \leq |X|, \\ 1 \leq j \leq 2|X|}} \bigvee_{0 \leq k \leq m} x[2,i,j,k], \tag{4}$$

$$\bigwedge_{\substack{1 \leq i \leq |X|, \\ 1 \leq j \leq 2|X|, \\ 0 \leq k[1] < k[2] \leq m}} (\neg x[2,i,j,k[1]] \vee \neg x[2,i,j,k[2]]), \tag{5}$$

$$\bigwedge_{1 \leq i \leq |X|} \bigvee_{\substack{1 \leq j \leq |X|, \\ 1 \leq k \leq 2|X|}} x[3,i,j,k], \tag{6}$$

$$\bigwedge_{\substack{1 \leq i \leq |X|, \\ 1 \leq j[1] \leq j[2] \leq |X|, \\ 1 \leq k[1] < k[2] \leq 2|X|}} (\neg x[3,i,j[1],k[1]] \vee \neg x[3,i,j[2],k[2]]), \tag{7}$$

$$\bigwedge_{\substack{1 \leq i \leq |X|, \\ 1 \leq j[1] < j[2] \leq |X|, \\ 1 \leq k[1] \leq k[2] \leq 2|X|}} (\neg x[3,i,j[1],k[1]] \vee \neg x[3,i,j[2],k[2]]), \tag{8}$$

$$\bigwedge_{\substack{1 \leq i[1] < i[2] \leq |X|, \\ 1 \leq j[2] < j[1] \leq |X|, \\ 1 \leq k[1] \leq 2|X|, \\ 1 \leq k[2] \leq 2|X|}} (\neg x[3,i[1],j[1],k[1]] \vee \neg x[3,i[2],j[2],k[2]]), \tag{9}$$

$$\bigwedge_{\substack{1\le i[1]<i[2]\le|X|,\\ 1\le j\le|X|,\\ 1\le k[2]<k[1]\le 2|X|}} (\neg x[3,i[1],j,k[1]] \vee \neg x[3,i[2],j,k[2]]), \quad (10)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le k\le 2|X|}} (x[1,j] \vee \neg x[3,i,j,k]), \quad (11)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le k\le 2|X|,\\ 0\le s\le m,\\ X[i]\neq a_s}} (\neg x[2,j,k,s] \vee \neg x[3,i,j,k]), \quad (12)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le k\le 2|X|,\\ 0< s\le m}} (\neg x[2,j,k,s] \vee x[3,i,j,k]). \quad (13)$$

The functions (4) – (13) allow us to select substrings in a partition of X. It is assumed that

$$P_i[j] = a_k$$

if and only if

$$x[2,i,j,k] = 1.$$

It is easy to see that $|P| \le |X|$. Let

$$y[1,1], \quad (14)$$

$$\neg y[1,|X|+1], \quad (15)$$

$$\bigwedge_{1\le i\le|X|} (y[1,i] \vee \neg y[1,i+1]). \quad (16)$$

We use the functions (14) – (16) to select $|P|$. In particular, we assume that

$$|P| = i$$

if and only if

$$y[1,i] = 1$$

and

$$y[1,i+1] = 0.$$

Let

$$\bigwedge_{1\le i\le|X|} \bigvee_{1\le j\le m} y[2,i,j], \quad (17)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j[1]<j[2]\le m}} (\neg y[2,i,j[1]] \vee \neg y[2,i,j[2]]). \quad (18)$$

We use the functions (17) and (18) to select P. It is assumed that

$$P[i] = a_j$$

if and only if

$$y[2,i,j] = 1.$$

Let

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|}} (\neg x[1,i+1] \vee \neg y[1,j] \vee (\bigvee_{1\le k\le 2|X|} y[3,i,j,k])), \quad (19)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le k\le 2|X|}} (y[1,j] \vee \neg y[3,i,j,k]), \quad (20)$$

$$\bigwedge_{1\le i\le|X|} (y[4,i] \vee \neg y[4,i+1]), \quad (21)$$

$$\bigwedge_{1\le i\le|X|} (y[1,i] \vee \neg y[4,i]), \quad (22)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|}} (\neg x[1,i] \vee x[1,i+1] \vee \neg y[4,j] \vee (\bigvee_{1\le k\le 2|X|} y[3,i,j,k])), \quad (23)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le k\le 2|X|}} (\neg x[1,i] \vee x[1,i+1] \vee y[4,j] \vee \neg y[3,i,j,k]), \quad (24)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le k[1]<k[2]\le 2|X|}} (\neg y[3,i,j,k[1]] \vee \neg y[3,i,j,k[2]]), \quad (25)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j[1]<j[2]\le|X|,\\ 1\le k[2]\le k[1]\le 2|X|}} (\neg y[3,i,j[1],k[1]] \vee \neg y[3,i,j[2],k[2]]). \quad (26)$$

The functions (19) – (26) allow us to select an alignment for P. Let

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le s\le 2|X|,\\ 0\le t[1]\le m,\\ 0\le t[2]\le m,\\ 0\le p\le d}} (\neg y[1,j] \vee \neg x[2,i,s,t[1]] \vee \neg y[3,i,j,s] \vee \neg y[2,j,t[2]] \vee z[1,i,s,p] = w[t[1],t[2],p]), \quad (27)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le s\le 2|X|,\\ 0\le t\le m,\\ 0\le p\le d}} \bigvee_{1\le j\le|X|} (y[3,i,j,s] \vee \neg x[1,i+1] \vee \neg x[2,i,s,t] \vee z[1,i,s,p] = w[t,0,p]), \quad (28)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le j\le|X|,\\ 1\le s\le 2|X|,\\ 0\le t[1]\le m,\\ 0\le t[2]\le m,\\ 0\le p\le d}} (\neg x[1,i] \vee x[1,i+1] \vee \neg y[4,j] \vee \neg x[2,i,s,t[1]] \vee \neg y[3,i,j,s] \vee \neg y[2,j,t[2]] \vee z[1,i,s,p] = w[t[1],t[2],p]), \quad (29)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le s\le 2|X|,\\ 0\le t\le m,\\ 0\le p\le d}} \bigvee_{1\le j\le|X|} (y[3,i,j,s] \vee \neg x[1,i] \vee x[1,i+1] \vee \neg x[2,i,s,t] \vee z[1,i,s,p] = w[t,0,p]). \quad (30)$$

We use the functions (27) – (30) to check distances between letters in the alignment. Let

$$\bigwedge_{\substack{1\le i\le|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee z[1,i,1,p] = z[2,i,1,p]), \quad (31)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1\le s\le 2|X|}} (\neg x[1,i] \vee \neg u[i,s,0]), \quad (32)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge u[i,s,p]) \to z[2,i,s,p])), \quad (33)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge u[i,s,p]) \to u[i,s,p+1])), \quad (34)$$

$$\bigwedge_{\substack{1\le i\le|X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge u[i,s,p]) \to \neg z[2,i,s,p])), \quad (35)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge u[i,s,p]) \rightarrow u[i,s,p+1])), \quad (36)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge u[i,s,p]) \rightarrow \neg z[2,i,s,p])), \quad (37)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge u[i,s,p]) \rightarrow u[i,s,p+1])), \quad (38)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow \neg z[2,i,s,p])), \quad (39)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow u[i,s,p+1])), \quad (40)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge u[i,s,p]) \rightarrow z[2,i,s,p])), \quad (41)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge u[i,s,p]) \rightarrow \neg u[i,s,p+1])), \quad (42)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow z[2,i,s,p])), \quad (43)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow \neg u[i,s,p+1])), \quad (44)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow z[2,i,s,p])), \quad (45)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow \neg u[i,s,p+1])), \quad (46)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge \neg u[i,s,p]) \rightarrow \neg z[2,i,s,p])), \quad (47)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 1<s\le 2|X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[1,i,s,p] \wedge \neg z[2,i,s-1,p] \wedge \neg u[i,s,p]) \to \neg u[i,s,p+1])). \quad (48)$$

The functions (31) – (48) allow us to check values of $\delta(P,P_i)$ and $\delta(P',P_r)$, for all $1 \le i < r$. Let

$$\bigwedge_{1\le i\le |X|} (\neg x[1,i] \vee \neg z[3,i,d+1]), \quad (49)$$

$$\bigwedge_{1\le i\le |X|} (\neg x[1,i] \vee z[4,i]), \quad (50)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[3,i,p+1] \wedge z[2,i,2|X|,p] > w[p]) \to \neg z[4,i]), \quad (51)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[3,i,p+1] \wedge z[2,i,2|X|,p] > w[p]) \to z[3,i,p]), \quad (52)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[3,i,p+1] \wedge z[2,i,2|X|,p] < w[p]) \to z[3,i,p]), \quad (53)$$

$$\bigwedge_{\substack{1\le i\le |X|,\\ 0\le p\le d}} (\neg x[1,i] \vee ((\neg z[3,i,p+1] \wedge z[2,i,2|X|,p] = w[p]) \to \neg z[3,i,p]). \quad (54)$$

To check the satisfiability of inequalities $\delta(P,P_i) < K$ and $\delta(P',P_r) < K$, for all $1 \le i < r$, we use the functions (49) – (54).

Let ξ be a conjunction of functions (1) – (54). It is not hard to verify that there is a string $U \in \Gamma^*$ such that U is a K-approximate period of X if and only if ξ is satisfiable.

Note that

$$\alpha \Leftrightarrow (\alpha \vee \beta_1 \vee \beta_2) \wedge (\alpha \vee \neg\beta_1 \vee \beta_2) \wedge (\alpha \vee \beta_1 \vee \neg\beta_2) \wedge (\alpha \vee \neg\beta_1 \vee \neg\beta_2), \quad (55)$$

$$\alpha_1 \vee \alpha_2 \Leftrightarrow (\alpha_1 \vee \alpha_2 \vee \beta) \wedge (\alpha_1 \vee \alpha_2 \vee \neg\beta), \quad (56)$$

$$\bigvee_{j=1}^{4} \alpha_j \Leftrightarrow (\alpha_1 \vee \alpha_2 \vee \beta_1) \wedge (\neg\beta_1 \vee \alpha_3 \vee \alpha_4), \quad (57)$$

$$\bigvee_{j=1}^{l} \alpha_j \Leftrightarrow (\alpha_1 \vee \alpha_2 \vee \beta_1) \wedge (\bigwedge_{i=1}^{l-4} (\neg\beta_i \vee \alpha_{i+2} \vee \beta_{i+1})) \wedge (\neg\beta_{l-3} \vee \alpha_{l-1} \vee \alpha_l), \quad (58)$$

$$\alpha = 1 \Leftrightarrow \alpha, \quad (59)$$

$$\alpha = 0 \Leftrightarrow \neg\alpha, \quad (60)$$

$$\beta \vee \alpha_1 = \alpha_2 \Leftrightarrow (\alpha_1 \vee \neg\alpha_2 \vee \beta) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \beta), \quad (61)$$

$$\bigwedge_{j=1}^{l} \alpha_j \to \beta \Leftrightarrow \beta \vee (\bigvee_{j=1}^{l} \neg\alpha_j), \quad (62)$$

$$\alpha < 1 \Leftrightarrow \neg\alpha, \quad (63)$$

$$\alpha > 0 \Leftrightarrow \alpha. \quad (64)$$

Using relations (55) – (64) one can easily obtain an explicit transformation of ξ into τ such that $\xi \Leftrightarrow \tau$ and τ is a 3-CNF. Clearly, τ gives an explicit reduction from AP to 3SAT.

IV. Experimental setup

In our experiments, we have used heterogeneous cluster (500 calculation nodes, Intel Core i7). Each test was runned on a cluster of at least 100 nodes. Due to restrictions on computation time (20 hours) we have used savepoints.

To obtain optimal solutions of AP we have used genetic algorithms OA[1] (see [12]), OA[2] (see [13]), OA[3] (see [14]), and OA[4] (see [15]) for the satisfiability problem. Also, we have considered fgrasp (see [16]), posit (see [16]), and GSAT with adaptive score function (see [17]) to solve the satisfiability problem.

V. Experimental results for DNA data

We use real world data from EMBL – EBI database [18]. We consider eukaryotic DNA sequences. In particular, we have created three test sets,

- $Test[1]$: substrings of 10^6 – $2 \cdot 10^6$ base pairs of DNA sequences of $2 \cdot 10^7$ – $21 \cdot 10^7$ base pairs (e.g. *Anopheles gambiae*, *Callithrix jacchus*);
- $Test[2]$: substrings of 10^5 – $2 \cdot 10^5$ base pairs of DNA sequences of 10^6 – 10^7 base pairs (e.g. *Candida dubliniensis*, *Dictyostelium discoideum*);
- $Test[3]$: substrings of 10^4 base pairs of DNA sequences.

Selected experimental results are given in Tables I – III.

TABLE I
EXPERIMENTAL RESULTS FOR $Test[1]$ DNA DATA

solver	average time	max time	best time
fgrasp	19.44 hr	85.72 hr	4.12 min
posit	17.61 hr	90.53 hr	7.54 min
GSAT	14.5 hr	74.33 hr	3.22 min
OA[1]	12.26 hr	61.06 hr	14.7 min
OA[2]	3.57 hr	71.25 hr	11.96 min
OA[3]	1.23 hr	21.47 hr	8.27 min
OA[4]	56.2 min	19.48 hr	6.19 min

TABLE II
EXPERIMENTAL RESULTS FOR $Test[2]$ DNA DATA

solver	average time	max time	best time
fgrasp	7.21 hr	36.2 hr	3.2 min
posit	6.43 hr	41.8 hr	4.6 min
GSAT	3.11 hr	74.33 hr	1.1 min
OA[1]	2.88 hr	19.37 hr	4.88 min
OA[2]	41.32 min	22.18 hr	6.59 min
OA[3]	19.7 min	6.3 hr	3.08 min
OA[4]	12.54 min	2.19 hr	2.4 min

VI. SEQUENCES OF MOTOR PRIMITIVES

Control systems of robots can allow quick use of motor primitives. The problem of learning of motor primitives is well studied. It is possible to rapidly learn motor primitives for many different complex behaviors, tennis-like swings [19], T-ball batting [20], drumming [21], biped locomotion [22], ball-in-a-cup [23], industrial applications [24]. However, ability to quickly and reliably use of different motor primitives is insufficient for fast adaptation to variations of the situation. Robots need an ability to select proper sequences of motor primitives. In particular, robots need to generalize motor primitives to a different behavior by trial and error without re-learning the task. In some cases, motor primitives can be adapted both spatially and temporally without changing the overall shape of the motion [19]. In particular, we can define meta-parameters as some small set of parameters that adapt the global motion behavior. A generalization of behaviors can be achieved by adapting these meta-parameters. For instance, the end position can be considered as a meta-parameter. Such approach was considered in the context of supervised learning for tennis-like swings with static ball targets [19], object manipulation [25], minigolf [26], drumming [27]. Also, supervised learning used to generalize meta-parameters in real-time [28]. Also, such approach was used in the context of reinforcement learning [29]. A prediction of a trajectory gives us another example of an adaptation of movement to situations. In particular, a prediction of a trajectory from a previously demonstrated set and refinement of this trajectory by motion planning is used for an adaptation of movement [30]. It should be noted that meta-parameters can be used to adapt to changes in the behavior of the robot itself. For instance, quadrotor vehicles are inherently unstable nonlinear systems. They exhibit exceedingly complex behavior at high speeds. An algorithm that exploits data from previous repetitions in order to learn to precisely follow a predefined trajectory was presented for quadrotor vehicles [31].

It is clear that sequences of motor primitives we can consider as strings in the finite alphabet of motor primitives. Let X be a sequence of motor primitives. If there is a string

TABLE III
EXPERIMENTAL RESULTS FOR $Test[3]$ DNA DATA

solver	average time	max time	best time
fgrasp	1.49 hr	8.88 hr	1.78 min
posit	52.82 min	9.03 hr	2.24 min
GSAT	24.9 min	2.62 hr	4.5 sec
OA[1]	21.3 min	1.88 hr	53 sec
OA[2]	3.02 min	43.6 min	35.2 sec
OA[3]	1.69 min	5.5 min	9.87 sec
OA[4]	48 sec	52.4 sec	36.1 sec

Fig. 1. Robot Neato XV-11 with an onboard computer and a camera.

U in the alphabet of motor primitives such that U is a K-approximate period of X for some relatively small integer K, then we can consider U as a meta-parameter. It is easy to see that such meta-parameters can be useful for the formulation of learning tasks for mobile robots. In particular, robots can use such meta-parameters for learning to generalize motor primitives to a different behavior by trial and error without re-learning the task and to predict a trajectory from a previously demonstrated sets.

In our experiments, we consider Neato XV-11 [32] with an onboard computer and a camera (see Figure 1). We have created two test sets,

- $Test[4]$: strings of 10^6 – $2 \cdot 10^6$ motor primitives;
- $Test[5]$: strings of 10^4 motor primitives.

Selected experimental results for SAT-solvers for sequences of motor primitives are given in Tables IV, V.

TABLE IV
SAT-SOLVERS FOR $Test[4]$ SEQUENCES OF MOTOR PRIMITIVES

solver	average time	max time	best time
OA[1]	3.11 hr	5.83 hr	56 sec
OA[2]	44.2 min	6.38 hr	48.01 sec
OA[3]	16.3 min	28.9 min	11.8 sec
OA[4]	8.02 min	12.94 min	16.7 sec

TABLE V
SAT-SOLVERS FOR $Test[5]$ SEQUENCES OF MOTOR PRIMITIVES

solver	average time	max time	best time
OA[1]	37.6 sec	59.26 sec	9.09 sec
OA[2]	24.2 sec	2.17 min	13.4 sec
OA[3]	7.03 sec	29.39 sec	0.26 sec
OA[4]	1.69 sec	3.55 sec	0.31 sec

It is natural to consider some recurrent neural network for learning to generalize motor primitives to a different behavior

by trial and error without re-learning the task. In particular, we can train some recurrent neural network N such that if X is a sequence of motor primitives, then $N(X)$ is a correction sequence of motor primitives. In general, $N(X)$ tries to choose the proper sequence from the set of all sequences. In this case, there are no preferences for the choice of the sequence.

We can use a genetic algorithm for training a recurrent neural network. In particular, we can consider a set of known pairs (A, B), where A is a learned task and B is a correction sequence of motor primitives for A. We can assume that if the distance between X and A is relatively small, then the distance between B and a correction sequence of motor primitives for X is relatively small. Let $f(Y)$ be the value of fitness function for Y. We assume that if the distance between X and A is relatively small and $\delta(Y_1, B) > \delta(Y_2, B)$, then $f(Y_1) < f(Y_2)$. Also, we consider Hamming distance. Let N_{GE} be a recurrent neural network that uses a genetic algorithm with δ. Let N_{GH} be a recurrent neural network that uses a genetic algorithm with Hamming distance.

Let U be a K-approximate period of a sequence of motor primitives X. In this case, we can represent X as $P_1 \dots P_r$, where $P_i \neq \epsilon$, $r > 2$, $\delta(U, P_i) < K$ for $1 \leq i < r$, and $\delta(U', P_r) < K$ where U' is some prefix of U. Let $C(U, P_i)$ be a correction sequence of motor primitives, where U is the learned task and P_i is a new task. Let

$$C(U) = \{C(U, P_i) \mid 1 \leq i < r\}.$$

We can consider $C(U)$ as the set of preferred correction sequences for U. We can use a genetic algorithm for training a recurrent neural network N_{GP} such that if $Y \in C(X)$ and $Z \notin C(X)$, then $f(Y) > f(Z)$.

Let $\mathrm{Trial}(S(X))$ be the number of trials that needed for recurrent neural network S to correct X. Let $\mathrm{Trial}(S)[t]$ be the average value of $\mathrm{Trial}(S(X))$ for strings of t motor primitives. Let

$$T(S, t) = \frac{\mathrm{Trial}(S)[t]}{\mathrm{Trial}(N)[t]}.$$

It is clear that we can consider values of $T(S, t)$ as a measure of the quality of recurrent neural network S. Selected experimental results are given in the Table VI.

TABLE VI
EXPERIMENTAL RESULTS FOR DIFFERENT RECURRENT NEURAL NETWORKS

t	10^3	10^4	10^5	10^6
$T(N_{GH}, t)$	0.56	0.43	0.28	0.27
$T(N_{GE}, t)$	0.42	0.35	0.23	0.22
$T(N_{GP}, t)$	0.14	0.05	0.012	0.003

It is clear that N_{GP} gives us the best results. However, X can be essentially aperiodic. In this case, we can find a K-approximate period of X only for relatively large values of K or for relatively small values of r. It is natural that for larger value of K we need larger number of trials. Also, it is clear that for relatively small values of r we need to consider very large U. So, it is natural to consider experimental results for N_{GP} for different values of K and $|U|$ (see Table VII).

TABLE VII
EXPERIMENTAL RESULTS FOR N_{GP} WITH DIFFERENT K AND $|U|$

t	10^3	10^4	10^5	10^6
$K < 3$, $\lvert U\rvert < 20$	0.02	0.001	0.0008	0.0002
$K < 5$, $\lvert U\rvert < 30$	0.03	0.006	0.0013	0.0004
$K < 9$, $\lvert U\rvert < 50$	0.07	0.03	0.005	0.001
$K \geq 9$, $\lvert U\rvert < 50$	0.36	0.24	0.09	0.03
$\lvert U\rvert \geq 50$	0.39	0.31	0.18	0.15

VII. CONCLUSION

In this paper, we have considered the approximate period problem. In particular, we have proposed an explicit reduction from the problem to the satisfiability problem. Also, we have presented experimental results for different satisfiability algorithms. Our experiments show that our explicit reduction to the satisfiability problem can be used for solution of the approximate period problem. It is clear that genetic algorithms demonstrate better performance than local search algorithms.

We have considered the approximate period problem for sequences of motor primitives of robots. In particular, we have used the approximate period problem to obtain some meta-parameters that adapt the global motion behavior. We have used such meta-parameters for learning to generalize motor primitives to a different behavior by trial and error without re-learning the task.

REFERENCES

[1] A. Gorbenko and V. Popov, "Self-Learning Algorithm for Visual Recognition and Object Categorization for Autonomous Mobile Robots," *Lecture Notes in Electrical Engineering*, vol. 107, pp. 1289-1295, January 2012.

[2] V. Popov, "Multiple genome rearrangement by swaps and by element duplications," *Theoretical Computer Science*, vol. 385, no. 1-3, pp. 115-126, October 2007.

[3] V. Yu. Popov, "Computational complexity of problems related to DNA sequencing by hybridization," *Doklady Mathematics*, vol. 72, no. 1, pp. 642-644, July-August 2005.

[4] J. S. Sim, C. S. Iliopoulos, K. Park, and W. F. Smyth, "Approximate periods of strings," *Theoretical Computer Science*, vol. 262, no. 1-2, pp. 557-568, July 2001.

[5] V. Popov, "The approximate period problem for DNA alphabet," *Theoretical Computer Science*, vol. 304, no. 1-3, pp. 443-447, July 2003.

[6] V. Popov, "The Approximate Period Problem," *IAENG International Journal of Computer Science*, vol. 36, no. 4, pp. 268-274, November 2009.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[8] A. Gorbenko, M. Mornev, and V. Popov, "Planning a Typical Working Day for Indoor Service Robots," *IAENG International Journal of Computer Science*, vol. 38, no. 3, pp. 176-182, August 2011.

[9] A. Gorbenko and V. Popov, "The set of parameterized k-covers problem," *Theoretical Computer Science*, vol. 423, no. 1, pp. 19-24, March 2012.

[10] A. Gorbenko and V. Popov, "Programming for Modular Reconfigurable Robots," *Programming and Computer Software*, vol. 38, no. 1, pp. 13-23, January 2012.

[11] A. Gorbenko, V. Popov, and A. Sheka, "Localization on Discrete Grid Graphs," *Lecture Notes in Electrical Engineering*, vol. 107, pp. 971-978, January 2012.

[12] A. Gorbenko and V. Popov, "SAT Solvers for the Problem of Sensor Placement," *Advanced Studies in Theoretical Physics*, vol. 6, no. 25, pp. 1235-1238, November 2012.

[13] A. Gorbenko and V. Popov, "Task-resource Scheduling Problem," *International Journal of Automation and Computing*, vol. 9, no. 4, pp. 429-441, August 2012.

[14] V. Popov, "Genetic Algorithms with Exons and Introns for the Satisfiability Problem," *Advanced Studies in Theoretical Physics*, vol. 7, no. 8, pp. 355-358, January 2013.
[15] V. Popov, "A Genetic Algorithm with Expansion Operator for the 3-Satisfiability Problem," *Advanced Studies in Theoretical Physics*, vol. 7, no. 8, pp. 359-361, January 2013.
[16] SATLIB — The Satisfiability Library. [Online]. Available: http://people.cs.ubc.ca/∼hoos/SATLIB/index-ubc.html
[17] V. Popov, "GSAT with Adaptive Score Function," *Advanced Studies in Theoretical Physics*, vol. 7, no. 8, pp. 363-366, January 2013.
[18] Genomes Pages - Eukaryota. [Online]. Available: http://www.ebi.ac.uk/genomes/eukaryota.html
[19] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in neural information processing systems, 2003*, pp. 1523-1530.
[20] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682-697, August 2008.
[21] D. Pongas, A. Billard, and S. Schaal, "Rapid synchronization and accurate phase-locking of rhythmic motor primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005*, pp. 2911-2916.
[22] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robotics and Autonomous Systems*, vol. 47, no. 23, pp. 79-91, March 2004.
[23] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171-203, February 2011.
[24] H. Urbanek, A. Albu-Schäffer, and P. van der Smagt, "Learning from demonstration repetitive movements for autonomous service robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004*, pp. 3495-3500.
[25] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation, 2009*, pp. 1293-1298.
[26] K. Kronander, M. S. Khansari-Zadeh, and A. Billard, "Learning to control planar hitting motions in a minigolf-like task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011*, pp. 710-717.
[27] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800-815, May 2010.
[28] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *IEEE International Conference on Robotics and Automation, 2011*, pp. 3719-3726.
[29] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361-379, April 2012.
[30] N. Jetchev and M. Toussaint, "Trajectory prediction: learning to map situations to robot trajectories," in *Proceedings of the 26th Annual International Conference on Machine Learning, 2009*, pp. 449-456.
[31] A. P. Schoellig, F. L. Mueller, and R. D'Andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1-2, pp. 103-127, February 2012.
[32] Neato Robotics web page. [Online]. Available: http://www.neatorobotics.com/

Anna Gorbenko was born on December 28, 1987. She received her M.Sc. in Computer Science from Department of Mathematics and Mechanics of Ural State University in 2011. She is currently a graduate student at Mathematics and Computer Science Institute of Ural Federal University and a researcher at Department of Intelligent Systems and Robotics of Mathematics and Computer Science Institute of Ural Federal University. She has (co-)authored 4 books and 90 papers.