Molecular Biology, Vol. 35, No. 6, 2001, pp. 841–848. Translated from Molekulyarnaya Biologiya, Vol. 35, No. 6, 2001, pp. 987–995.

UDC 577.1

Algorithm of Regulatory Signal Recognition in DNA Sequences

L. V. Danilova[1], K. Yu. Gorbunov[1], M. S. Gelfand[2], and V. A. Lyubetskii[1]

[1] *Institute of Problems of Data Transmission, Russian Academy of Sciences, Moscow, 101447 Russia; E-mail: dlv2k@mail.ru*

[2] *State Research Center of Genetics, Moscow, 113545 Russia*

Received November 17, 2000

Abstract—An algorithm is proposed for extracting regulatory signals from DNA sequences. The algorithm complexity is nearly quadratic. The results of testing the algorithm on artificial and natural sequences are presented.

Key words: molecular biology, regulatory signal, computer logic, graph algorithms

INTRODUCTION

The problem of regulatory signal recognition is a classical problem of computational biology. It has become particularly popular within the recent years, after, on the one hand, initiation of mass experiments on gene expression analysis (e.g., [1, 2]; reviewed in [3, 4]) and, on the other hand, publication of complete genomes of many bacteria and some eukaryotes, which allowed comparative analysis of the control processes ([5, 6]; reviewed in [7]). Despite its importance, this problem is far from being solved. Accurate (that is, exhaustion) methods are so time-consuming as to be hardly practicable with real tasks.

The available approaches and algorithms for recognizing regulatory signals in a set of potential regulatory regions have been reviewed [8–11].

These algorithms can be divided into two groups: optimization and combinatorial. Optimization algorithms are organized as follows. First, a quality (e.g., information content) of sets of the potential sites is described. Then such sets are generated, and selection of representatives in each sequence is gradually refined to maximize the quality. Thus, the entire procedure is reduced to the search of an extremum of the quality function in the space of sets. In this case greedy algorithms [12, 13], expectation maximizing algorithms [14–17], DMS [18], MEME [19–21], as well as stochastic algorithms of heat annealing simulation [22] and Gibbs sampler [23, 24] can be applied.

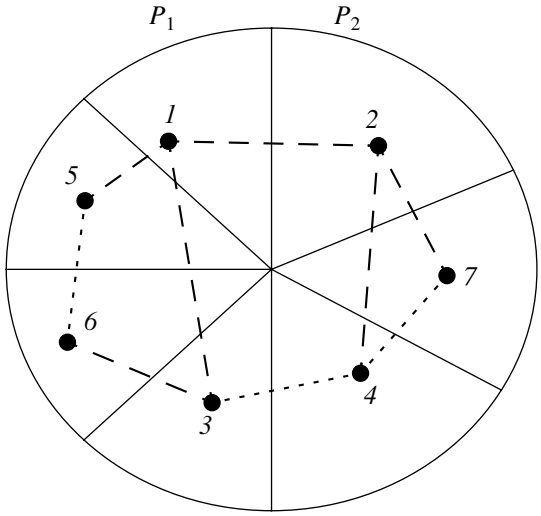
Exhaustion algorithms also work with word sets; however, in this case they aim at building a word present in each sequence in the sample with least deviation (i.e., the quality function is a measure of compactness of the generated set, such as its diameter). Such algorithms include ConsInd and MatInd [25, 26]; ITB [27]; WORDUP [28, 29]; CONSENSUS [30]; WINNOWER [31]; suffix trees [32–34]; and other algorithms [35–43].

Here we propose and test a **new algorithm for signal recognition in a set of unaligned nucleotides sequences** (see also [44]). It is intermediate in terms of the above classification: a quality function is optimized, but it is defined as a pairwise similarity between words rather than as information content of a set.

ALGORITHM

Problem statement. We are given a set of k nucleotide sequences, the length of which is not fixed or is about the same (in the latter case it equals certain n in each case). Let *system* denote a set of words of fixed length l, one word per sequence (or several words per sequence; here we consider the first case for shortness); the system includes a set of some *not predefined part of the initial sequences*; the system should be composed of the most pairwise-similar words (and of the largest number of the sequences). We consider similarity as, e.g., total pairwise Hamming distance between the words included in the system or as some other fixed metric between these words or, stated differently, as maximization of a fixed "quality of the system." The quality of the system is defined as, e.g., total pairwise "distance" between its words calculated from function $F(x, y)$ reflecting the similarity between words x and y of length l as well as other desired properties in such a pair of words. Intuitively, such system

0026-8933/01/3506-0841$25.00



The procedure of graph *G* partitioning. Partitioning of graph *G* at $k = 7$ is exemplified. *Step 1*: graph *G* is divided into two parts P_1 and P_2 so that $P_1 = \{1, 3, 5, 6\}$ and $P_2 = \{2, 4, 7\}$, $(1, 2)$ and $(3, 4)$ are the main and auxiliary edges of this division, respectively; *step 2*: part P_1 is divided into $P_{11} = \{1, 5\}$ and $P_{12} = \{3, 6\}$ with $(1, 3)$ and $(5, 6)$ as the main and auxiliary edges, respectively, while part P_2 is divided into $P_{21} = \{2, 7\}$ and $P_{22} = \{4\}$ with $(2, 4)$ and $(4, 7)$ as the main and auxiliary edges, respectively; *step 3*: part P_{11} is divided into $P_{111} = \{1\}$ and $P_{112} = \{5\}$ with $(1, 5)$ as the main edge, part P_{12} is divided into $P_{121} = \{3\}$ and $P_{122} = \{6\}$ with $(3, 6)$ as the main edge, and part P_{21} is divided into $P_{211} = \{2\}$ and $P_{212} = \{7\}$ with $(2, 7)$ as the main edge.

can be considered as a *signal*, while a word representing the signal in one of the source sequences can be considered as a *regulatory region (site)* in this sequence.

A serious algorithmic and theoretic problem is the possible absence of regulatory sites related to the considered signal from some sequences of the initial sample. Hence, it is important that the described algorithm can find the signal present in a relatively small fraction of the initial sequences.

If necessary, the structural properties of the signal can be considered. For instance, if there are grounds to believe that the signal is a (complementary) palindrome (in particular, if the regulatory protein is known to bind DNA as a dimer), one can use function $F(x, y) = S(x, y) + 0.5\,[\max(S(x, pal(x)), S(x', pal(x'))) + \max(S(y, pal(y)), S(y', pal(y')))]$, where $S(x, y)$ is the number of matching characters in words x and y, $pal(x)$ is the word derived from x by inversion of each character to the complementary one, and x' is the word x without the last character. This weighting function was largely selected at random, and the algorithm is independent of the function F type except that the extreme complexity of its calculation clearly increases the algorithm calculation time. However, there are no grounds to expect biologically informative and hard-to-calculate functions F that could considerably affect the calculation time of the algorithm.

The proposed algorithm solves the initial problem within the time of **squared number of the initial sequences *k* and cubed length *n* of each of them**. An algorithm with a better calculation time estimate is hardly imaginable.

Qualitative description of the algorithm. First let us form an auxiliary graph *G* fixed during the algorithm execution. Graph *G* is composed of *k* nodes and the edges appearing during the next procedure execution as exemplified in the figure where $k = 7$. At the first step all nodes of graph *G* are divided into two equal (accurate to unity for odd *k*) parts and two edges (A, B) and (C, D) not going beyond any node are laid (e.g., let *A* and *C* be in one part while *B* and *D* be in the other part). Each of these edges, e.g., (A, B) can be denoted as *the main relative to this division* while the other one is *auxiliary*. $(A, B) = (1, 2)$ and $(C, D) = (3, 4)$ in the figure. In addition two *diagonal* edges (A, D) and (C, B) are laid (not shown in figure). Later such division is iterated "inside" the graph *G*. Namely, each of the parts of graph *G* is redivided (in the same sense) into equal parts so that *A* and *C* as well as *B* and *D* fall in different parts of these divisions. The main edges are unambiguously defined relative to these new divisions, namely, (A, C) and (B, D), while the auxiliary edges (if possible not going beyond the same node as the main ones) are arbitrarily selected (edges $(5, 6)$ and $(4, 7)$ in figure). So forth each not single-node part *P* appeared in this procedure is divided into two equal parts P_1 and P_2 so that the main edges of the new parts link the termini of the main and auxiliary edges of the previous part *P*. Of course, in this case each *P* equals its merged parts P_1 and P_2.

The division procedure is over when all *P* parts are single-node; in reality it can be stopped when these parts are just small (1–3 nodes).

The essence of the main algorithm cycle can be described as follows: in the case of small parts head-on solutions (exhaustive search) are used for any problem, otherwise inductive transition from the calculated (available) data for P_1 and P_2 to homogeneous data on *P* is used. Stated differently, *f* data for P_1 and *g* for P_2 can be used for easy calculation of homogeneous (as compared to *f* and *g*) *h* for *P*. Of course, this is just a brief description detailed below.

The external cycle of the algorithm is one-to-one assignment of the source sequences to the nodes of graph *G*. Let us denote one of such (current but fixed for each individual iteration of the cycle) arrangements of sequences by the nodes of graph *G* as *r*. Then $r(A)$ is the sequence assigned to node *A* where the signal is searched as well as in all other assigned

sequences. Graph *G* helps to organize this search. This cycle body is organized as follows.

Let us call sequence *r(A)* a sequence *over A*.

A new cycle (called *assembly*) is executed for the fixed *r*; it performs *transition* from the data on two "neighboring" (and smaller) parts P_1 and P_2 of graph *G* to the data on their merging into a (larger) part *P* of graph *G* or, to be precise, transition from the data on merged sequences over *B* (where *B* runs P_2 part) to the data on merged sequences over *C* (where *C* runs *P* part).

Let *sequences over P* denote merged sequences *r*(*A*), where *A* runs *P* part. The system of words (no more than) one per each sequence over *P* is called a *signal (system of words) over P.* Of course, we search for a signal over $P = G$ which is called the signal; however, we search this signal as a special case over an arbitrary *P.*

Finally, the best possible system of words (no more than one per each sequence over *P*) with the signal value over the main edge of *P* part equal to two predefined (arbitrary in the general case) words *x* and *y* from the corresponding sequences is called an *extension of words x and y to the signal over P.* Clearly, extension of some words *x* and *y* to the signal over *P* can be not a (sensible) signal over *P* due to these "fixed" *x* and *y* values.

Thus, the above-mentioned transition (inductive step) consists in the following.

Assume we have determined two sets of *t* best signals (systems) all over the parts P_1 and P_2, respectively; these parts with the main edges (*A, C*) and (*B, D*), respectively, were obtained by dividing subgraph *P* (of the initial graph *G*) with the main edge (*A, B*). Clearly, simple merging of the best signal over P_1 and the best signal over P_2 is not even a (sensible) signal over *P* (in the general case).

Hence we will use a more delicate solution. The above set, e.g., for P_1 consists of "best (possible) signals with fixed ends" over P_1 rather than of the best signals over P_1; i.e., the first set consists of extensions of any two words *x* and *y* from the sequences over *A* and *C*, respectively (where the mutual quality of *x* and *y* exceeds certain fixed *threshold u*), to the P_1 set. In a similar way, the second set is composed of *t* best extensions to the whole set P_2 of words *x* and *y* arbitrarily selected over nodes *B* and *D*.

For example (figure), let $P_1 = \{1, 5, 6, 3\}$ and $P_2 = \{2, 7, 4\}$; $(A, C) = (1, 3)$ and $(B, D) = (2, 4)$; while *P* equals merged sets P_1 and P_2 (coincidence of *P* and *G* in this case is clearly a random event; it always happens at the end of this induction rather than at the preceding steps). Let $t = 1$ (by the way, this was true for the biological cases used for the algorithm testing). At the previous iterations of the considered cycle two functions $f(x, y)$ and $g(x, y)$ were calculated. Function $f(x, y)$ on any pair of words (similarity between *x* and *y* from sequences over *A* and *C* exceeds certain fixed threshold *u*) outputs the best signal in the sample $\{r(1), r(3), r(5), r(6)\}$ corresponding to the P_1 set (at the fixed *r*). In a similar way, function $g(x, y)$ on any pair of words (*x* and *y* from sequences over *B* and *D* are similar enough in the same sense) outputs the best signal in the sample $\{r(2), r(4), r(7)\}$ corresponding to the P_2 set (at the same fixed *r*). This iteration underlies the building function $h(x, y)$ outputting analogous information for a larger set *P.*

Namely, *search for all words* x_1 and y_1 from the sequences over *C* and *D*, respectively, with the pairwise quality of words *x* and x_1 as well as *y* and y_1 above this threshold (to be precise, with defined extensions) allows us to select (using *f* and *g*, respectively) separate $x–x_1$ and $y–y_1$ extensions on P_1 and P_2 giving us *t* best signals over the whole *P* upon merging all *P.*

If the threshold condition cannot be satisfied, the corresponding value of the signal over *P* is set to zero; stated differently, the threshold condition can provide for partially defined signals over *G*, which is, however, natural in terms of statement of the initial problem.

In addition, a list *S* of the obtained signals is maintained. As soon as a next signal *s* appears it is tested for similarity to the signals already included in *S* (within the same determination area). This is done in the following way: if *s* signal has no new words as compared to relatively large fraction of signals from *S*, we assume that significant portion of these not new words from *s* are the real sites; in this case we evaluate how close is each of the words remained in *s* to this *signal* set. If no similar words have been found, this signal *s* is not included in *S*.

The external cycle arranging the sequences by graph *G* nodes provides for at least single assignment of each sequence pair to the nodes of graph *G* so that they are linked by an edge; in this case the next arrangement is selected to increase the number of linked sequence pairs not linked in previous iterations. This provides for sensible amount of iterations of the external cycle and sufficient diversity of the calculated arrangements *r.*

The latter is important for generating representative statistics during the next final cycle of the algorithm execution.

Namely, each position in each sequence (e.g., containing character *i*) is assigned to a measure of character *i* occurrence in the desired site. In the second variant such measure and each word are juxtaposed, which reflects the occurrence of this word in the desired site (below we consider this variant). This measure equals total quality by all revealed signals including this word. Here the quality states for quality of the word from the signal (containing it) relative to

the whole signal rather than quality of the whole signal, i.e., total $F(x, y)$, where x is the above word while y runs all other words of this signal. Thus, the words from the "real" signal will be marked by significantly higher values as compared to other words.

The algorithm idea is as follows. The notion of the best signal (pretending to be real) requires considering all sequence pairs (carrying the signal). The signals corresponding to the best signal definition are selected for each arrangement r for the sequence pairs linked by an edge in (the current arrangement of) graph G and the number of such edges is by far less than the number of all possible sequence pairs. However, selection of the appropriate arrangements chain allows us to obtain representative statistics for the subsequent signal recognition. This paragraph intuitively explains the idea of the algorithm; the proof problem is not discussed here.

IMPLEMENTATION OF THE ALGORITHM AND COMPUTATIONAL EXPERIMENT

The algorithm was implemented as a computer program mostly used for signal search in the sample of genetic sequences with representative length from 100 to 200 nucleotides; in this particular case the signal was a system of individual words with representative length from 15 to 30. In this variant the program was composed in object-oriented Pascal language in Delphi programming environment. The input was a text file containing a set of genetic sequences, and the output was a file containing the function relating each word with the confidence of its occurrence in the signal. A 300 MHz Celeron with 64 MB processed 19 and 9 sequences of length 200 in 12 and 1.2 min, respectively.

Calculation of many examples demonstrated that the algorithm suggested at least one site in each signal-carrying sequence in the great majority of cases.

We used the following procedure to test if a given sequence carries other signal words. Each character of the revealed word in a given sequence was replaced by asterisk sign considered by the program as different from all other characters including itself. After that the algorithm found another signal word (if present), etc.

In rare cases the algorithm found no signal word in a sequence (although it was there). This is due to the

Table 1. Results of the testing on artificial sequences

	0, 1, 2	3	4	5	6
0	XXXXXXXXXX	XXXXXXXXXX	XXXXXXXXXX	1061011555	1220021000
1	XXXXXXXXXX	XXXXXXXXXX	XXXXX9XXXX	0063	–
2	XXXXXXXXXX	XXXXXXXXXX	XXXX97XX7X	0053	–
3	XXXXXXXXXX	XXXXXXXXXX	X9X478XX9X	0065	–
4	XXXXXXXXXX	XXXXXXXXXX	799286X96X	1000	–
5	XXXXXXXXXX	XXXXXXXXXX	3498168858	0020	–
6	XXXXXXXXXX	XXXXXXXXXX	05X0477052	0000	–
7	XXXXXXXXXX	XXXXXXXXXX	0330053404	0000	–
8	XXXXXXXXXX	XXXXXXXXXX	2240004025	0000	–
9	XXXXXXXXXX	XXXXXXXX9X	2650427021	0000	–
10	XXXXXXXXXX	XXXXXXXXXX	8423552057	–	–
	0, 1	2	3	4	
0	5	5555555555	5555555555	554530255	
1	5	5555555555	5555555555	255413144	
2	5	5555555555	5555555555	514330023	
3	5	5555555555	5555453541	005000030	
4	5	5555555555	5535255554	003020011	
5	5	4555555555	3403535443	000300013	

Note: The number of correctly recognized sites is given, each character corresponds to single independent test (X states for 10), upper table, sample of ten sequences; lower table, sample of five sequences; rows indicate the number of sequences lacking the site added to the initial set (from 0 to 10 and from 0 to 5, respectively); columns indicate the number of changed characters (from 0 to 6 and from 0 to 4, respectively).

"garbage"—a set of quite similar but not signal words. Such garbage featured relatively high values and could be easily recognized (particularly, using biological considerations). After such "garbage" words were replaced by asterisks the program was re-executed and found not previously revealed signal words.

RESULTS AND DISCUSSION

Artificial sample. The algorithm was described by its testing on artificial sample under controlled conditions. The same word of length 16 was substituted into artificial sequences of length 200 in the four-letter code. Then several characters were "spoiled" in each of these words (to simulate attenuation of the signal) and sequences without the signal were added (to simulate sample pollution).

The results of testing are presented in Table 1.

The signals are stably found after introduction of up to two errors in the signal in a 5-sequence sample, up to three errors in a 10-sequence sample, and when the fraction of irrelevant sequences (lacking the site) was up to 50% of the sample. The 10-sequence case was studied in detail. In the case of errors in four positions in the signal the result depends on the sample purity: acceptable results can be obtained when the number of the sequences lacking the site do not exceed 3 or 4 (in most tests the sites were correctly recognized in virtually all sequences). The signal can be missed upon further pollution of the sample; the proportion of such tests increases with the number of irrelevant sequences. Only single tests reveal weaker signals.

Natural samples. An analogous approach was used to consider three sequence samples containing *Escherichia coli* regulatory sites. Sample pollution by the irrelevant sequences was stimulated as follows: the best site found at the current stage of testing was

Table 2. Recognition of PurR binding sites

	0	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12	11	13
Operon	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M
purR-1	88	8	*	8	*	8	*	14	*	7	*	6	*	6	*	6	*	13	*	7	*	**12**	*	**12**	*	7	*	7
purR-2	0		72		63		41		27		24		24		23		23		15		7		7		7		7	
purEK	88	0	88	0	*	12	*	7	*	18	*	12	*	12	*	19	*	12	*	7	*	7	*	7	*	7	*	7
CvpApurF	88	0	88	0	76	0	*	22	*	22	*	21	*	14	*	14	*	13	*	7	*	7	*	7	*	7	*	7
purC	87	0	80	0	70	0	59	0	29	0	18	6	18	6	22	6	24	7	15	7	15	7	8	7	8	7	7	7
purMN	88	0	88	0	70	0	60	0	30	0	28	0	*	22	*	42	*	13	*	13	*	13	*	6	*	7	*	12
purL	88	0	88	0	75	0	60	6	*	24	*	24	*	28	*	20	*	13	*	7	*	12	*	12	*	12	*	12
purB	80	0	80	0	59	7	45	6	27	6	17	6	17	6	16	6	16	7	13	7	13	7	8	7	8	7	**6**	**11**
guaBA	79	0	80	0	67	0	54	0	27	0	17	7	17	6	23	14	24	7	*	14	*	14	*	14	*	14	*	21
purHD	88	0	88	0	70	0	59	0	30	0	27	7	27	7	24	7	24	7	7	**13**	8	7	8	6	7	7	**0**	**14**
glyA	80	0	80	0	70	0	59	0	29	0	26	6	26	8	25	7	17	7	8	7	8	7	8	7	**8**	**12**	**0**	**8**
pyrD	88	0	88	0	70	0	49	0	29	0	26	0	26	0	*	12	*	12	*	7	*	7	*	7	*	7	*	11
prsA	88	0	88	0	70	0	60	0	30	0	*	28	*	21	*	19	*	7	*	12	*	7	*	7	*	7	*	12
glnB	80	0	80	0	63	0	53	0	27	0	16	8	15	8	14	7	7	6	13	6	13	6	13	6	*	6	*	7
purA-1	80	0	80	0	63	0	53	0	27	0	24	6	24	6	31	5	22	**6**	7	7	7	7	11	7	11	7	*	7
purA-2	0		0		0		0		0		0		0		7		**6**		**6**		**6**		**0**		**0**		**0**	
codBA	88	0	88	0	75	0	60	0	30	0	27	0	26	0	32	6	17	8	15	8	*	8	*	10	*	8	*	8
pyrC	80	0	80	0	69	0	59	0	29	0	18	5	18	5	9	7	24	7	15	6	7	7	7	6	7	7	7	7
purT	88	0	88	0	76	0	61	0	30	0	27	0	27	0	26	5	*	7	*	12	*	12	*	12	*	7	*	12
GcvTHP	72	0	72	0	63	0	45	7	26	7	15	0	14	7	14	6	15	7	15	6	15	6	7	6	7	6	7	7
speAB	70	8	70	8	54	5	45	6	18	5	17	8	17	8	23	5	16	7	15	7	15	7	*	7	*	11	*	13

Note: First row indicates the number of sequences lacking the site and total number of masked sites (indicated by asterisks); second row: C, weight of a true site; M, maximum weight of a false site; boldface marks missed true site or best false site with the value equal or higher than that of the best true one (except zero weight).

Table 3. Recognition of ArgR binding sites

	0	0	0	1	0	2	0	3	0	4	1	5	1	6	1	7	1	8	0	9
Operon	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M
argR-1	33	0	36	0	*	0	*	8	*	8	*	0	*	**13**	*	**18**	*	**9**	*	**18**
argR-2	0		0		24		34		18		8		**0**		**8**		**8**		**8**	
argA-1	33	0	32	0	19	0	19	0	17	8	9	7	9	8	**0**	**16**	**0**	**16**	*	**16**
argA-2	0		0		9		26		17		9		9		**9**		**0**		**9**	
argCBH-1	0	0	**2**	**6**	18	0	37	7	27	7	16	8	**8**	**8**	**8**	**8**	**0**	**9**	**10**	**18**
argCBH-2	39		*		*		*		*		*		*		*		*		*	
argD-1	0	0	0	0	0	0	0	0	8	7	7	7	**7**	**8**	**8**	**8**	**8**	**7**	**8**	**16**
argD-2	35		35		30		51		*		*		*		*		*		*	
argE-1	0	0	0	0	9	0	10	0	10	0	8	0	**0**	8	**0**	**9**	**0**	**9**	**10**	**18**
argE-2	39		27		22		43		38		8		9		**9**		**8**		*	
argF-1	36	0	36	0	31	0	21	0	19	0	**6**	**14**	10	8	*	**17**	*	**8**	*	**18**
argF-2	0		0		0		30		27		**6**		**0**		**7**		**7**		**10**	
argG-1	11	0	0	0	10	0	10	0	19	0	**8**	8	8	7	**9**	**16**	**0**	**9**	**10**	**16**
argG-2	0		0		0		0		0		**0**		8		**0**		**0**		**0**	
argG-3	24		36		22		42		29		10		*		*		*		*	
argI-1	36	0	35	0	31	0	*	0	*	0	*	15	*	8	*	16	*	8	*	**16**
argI-2	0		0		0		47		44		*		*		*		*		**10**	
carAB-1	30	0	16	0	24	0	18	0	19	0	9	8	9	8	17	8	*	**8**	*	**16**
carAB-2	0		7		0		9		8		**8**		**8**		**0**		**0**		**8**	

Note: For designations see Table 2. The best sites masked in each sequence in the last pair of columns.

masked (its nucleotides were replaced by asterisk signs). This both introduced the sequences lacking the signal and gradually weakened the signal.

The regulatory site signals were taken from the dpinteract database [45], while sequence fragments containing these sites were extracted from the complete *E. coli* genome using GenomeExplorer [46].

Purine regulon. The sample of the regulatory regions of the genes controlled by purine repressor PurR included 19 sequences 200 nucleotides long containing 21 sites 16 nucleotides long. Two sequences carried two sites while the other ones carried single site each. Table 2 presents the results of the testing. First errors appear when 8–9 sites are masked; however, even when the sample included more than a half sequences lacking the site, most sites were correctly recognized. When two sites are present in the same sequence, the second one is found after masking of the already revealed one. The purine sample contains two such sequences.

Arginine regulon. The sample of regulatory regions of the genes controlled by arginine repressor ArgR included 9 sequences 200 nucleotides long containing 19 sites 18 nucleotides long. One sequence contained three sites while the other ones contained two sites each. Table 3 presents the results of testing. Arginine box is a weak signal and specificity of the control is due to cooperative recognition of site pairs at a fixed distance by the multimeric repressor complexes [47]. Nevertheless, the arginine repressor binding sites is reliably found even when considerable number of the best sites was masked: the first loss is observed after masking five sites (two of them falling in the same sequence; the second one is found after masking the first one as with the purine sample). In the case of triple sites the procedure is the same: the third site is found after masking two previously found ones.

Catabolite repressor regulon. The sample of regulatory regions of the genes controlled by CRP included 31 sequences 2 hundred nucleotides long containing 48 sites 22 nucleotides long. Sixteen sequences carried single site while the other ones carried from 2 to 4 sites. The sample of CRP binding sites includes numerous weak sites; many of them have not been found even in the primary search (Table 4). That is why no tests with the sites masking were carried out. Note that the interaction between CRP and the

Table 4. Recognition of CRP binding sites

	C	M
aldB	**8**	**15**
ansB	**0**	**18**
araB-1	17	8
araB-2	**7**	
cdd-1	**0**	7
cdd-2	28	
crp-1	**0**	14
crp-2	16	
cya	27	9
cytR-1	17	16
cytR-2	7	
dadAX-1	43	8
dadAX-2	**8**	
deoP-1	**9**	**17**
deoP-2	**8**	
fur	26	13
gal	**8**	**21**
glpACB-1	26	8
glpACB-2	**8**	
glpACB-3	**8**	
glpD	17	8
glpFK-1	**0**	15
glpFK-2	32	
gut	34	14
ilvB	**10**	**17**
lac-1	9	7
lac-2	24	
malEpKp-1	8	6
malEpKp-2	9	
malEpKp-3	9	
malEpKp-4	**0**	
malT	18	16
melR	**16**	**30**
mtl	**17**	**21**
nupG-1	20	9
nupG-2	19	
ompA	**16**	**16**
ompR	24	16
ptsH-1	**9**	**26**
ptsH-2	**16**	
rhaS	19	8
rot-1	**8**	8
rot-2	27	
tdcA	28	8
tnaL	28	8
tsx-1	**10**	**26**
tsx-2	7	
uxuAB	23	8

Note: See Table 2 for designations.

regulatory regions is complex and includes dynamic switch between the sites [48].

Hence, it is possible that some of the erroneously recognized sites are actually functional.

CONCLUSIONS

The testing demonstrated eligibility of the proposed algorithm. Presently it is applied to study control interactions, particularly, it is involved in the analysis of unknown sugar metabolism regulons in γ-proteobacteria and systematic investigation of regulation in pyrococci.

Several trends of further development of the algorithm can be proposed. Increased resistance to noise and, particularly, to the sample pollution is the topical problem. We intend to modify the algorithm to explicitly consider possible pollution, while the expected number of irrelevant sequences should be a parameter set from *a priory* considerations or adjusted automatically.

In addition, statistical peculiarities of the studied genomes should be considered, particularly, heterogeneity of the nucleotide composition and the frequencies of oligonucleotides (a word nonrandom relative to the whole genome can hardly be a specific regulatory signal).

We should accurately investigate possible application of various types of symmetry (palindromes, director repeat, etc.). Finally, the algorithm should more specifically allow for possible appearance of several sites in a sequence as well as several different signals in a sample.

ACKNOWLEDGMENTS

We thank A.A. Mironov for the help with GenomeExplorer and valuable discussion. This work was supported in part by Merck Genome Research Institute (project no. 244), INTAS (project no. 99-1476), Russian Foundation for Basic Research (projects nos. 99-04-48247 and 00-15-99362), Howard Hughes Medical Institute (project no. 55000309), and the Russian program Human Genome.

REFERENCES

1. Spellman, P.T. *et al.*, *Mol. Biol. Cell*, 1998, vol. 9, pp. 3273–3297.
2. Roth, F.R. *et al.*, *Nature Biotechnol.*, 1998, vol. 16, pp. 939–945.
3. Bassett, D.E. Jr. *et al.*, *Nature Genet.*, 1999, vol. 21, pp. 51–55.
4. Bucher, P., *Curr. Opin. Struct. Biol.*, 1999, vol. 9, pp. 400–407.
5. Gelfand, M.S., Koonin, E.V., and Mironov, A.A., *Nucleic Acids Res.*, 2000, vol. 28, pp. 695–705.

6. McGuire, A.M., Hughes, J.D., and Church, G.M., *Genome Res.*, 2000, vol. 10, pp. 744–757.
7. Gelfand, M.S., *Res. Microbiol.*, 1999, vol. 150, pp. 755–771.
8. Gelfand, M.S., *J. Comput. Biol.*, 1995, vol. 2, pp. 87–115.
9. Frech, K., Quandt, K., and Werner, T., *Comput. Appl. Biosci.*, 1997, vol. 13, pp. 89–97.
10. Duret, L. and Bucher, P., *Curr. Opin. Struct. Biol.*, 1997, vol. 7, pp. 399–406.
11. Fickett, J.W. and Wasserman, W.W., *Curr. Opin. Biotechnol.*, 2000, vol. 11, pp. 19–24.
12. Stormo, G.D. and Hartzell, G.W. III., *Proc. Natl. Acad. Sci. USA*, 1989, vol. 86, pp. 1183–1187.
13. Hertz, G.Z., Hartzell, G.W. III, and Stormo, G.D., *Comput. Appl. Biosci.*, 1990, vol. 6, pp. 81–92.
14. Lawrence, C.E. and Reilly, A.A., *PROTEINS: Structure, Function, Genetics*, 1990, vol. 7, pp. 41–51.
15. Cardon, L.R. and Stormo, G.D., *J. Mol. Biol.*, 1992, vol. 223, pp. 159–170.
16. Frishman, D., Mironov, A., Gelfand, M., *Gene*, 1999, vol. 234, pp. 257–265.
17. Gelfand, M.S., Koonin, E.V., and Mironov, A.A., *Nucleic Acids Res.*, 2000, vol. 28, pp. 695–705.
18. Hu, Y.-J., Sandmeyer, S., McLaughlin, C., and Kibler, D., *Bioinformatics*, 2000, vol. 16, pp. 222–232.
19. Bailey, T.L. and Elkan, C.P., *Proc. 2nd Int. Conf. on Intelligent Systems for Molecular Biology ISMB'1994*, 1994, pp. 28–36.
20. Bailey, T.L. and Elkan, C.P., *Proc. 3rd Int. Conf. on Intelligent Sysems for Molecular Biology ISMB'1995*, 1995, pp. 21–29.
21. Grundy, W.N., Bailey, T.L., and Elkan, C.P., *Comput. Appl. Biosci.*, 1996, vol. 12, pp. 303–310.
22. Lukashin, A.V., Engelbrecht, J., and Brunak, S., *Nucleic Acids Res.*, 1992, vol. 20, pp. 2511–2516.
23. Lawrence, C.E., Altschul, S.F., Boguski, M.S., *et al.*, *Science*, 1993, vol. 262, pp. 208–214.
24. Roth, F.P., Hughes, D., Estep, P.W., and Church, G.M., *Nature Biotech.*, 1998, vol. 16, pp. 939–945.
25. Frech, K., Herrmann, G., and Werner, T., *Nuceic. Acids Res.*, 1993, vol. 21, pp. 1655–1664.
26. Quandt, K., Frech, K., Karas, H., *et al.*, *Nucleic Acids Res.*, 1995, vol. 23, pp. 4878–4884.
27. Kielbasa, Sz.M., Korbel, J.O., Beule D., *et al.*, *Proc. German Conf. Bioinformatics GCB'2000*, 2000, pp. 55–62.
28. Pesole, G., Prunella, N., Liuni S., *et al.*, *Nucleic Acids Res.*, 1992, vol. 20, pp. 2871–2875.
29. Liuni, S., Prunella, N., Pesole, G., *et al.*, *Comput. Appl. Biosci.*, 1993, vol. 9, pp. 701–707.
30. Hertz, G.Z. and Stormo, G.D., *Bioinformatics*, 1999, vol. 15, pp. 563–577.
31. Pevzner, P.A. and Sze, S.-H., *Proc. 8th Int. Conf. on Intelligent Systems for Molecular Biology ISMB'2000*, 2000, pp. 269–278.
32. Jonassen, I., *Comput. Appl. Biosci.*, 1997, vol. 13, pp. 509–522.
33. Brazma, A., Jonassen, I., Vilo, J., and Ukkonen, E., *Genome Res.*, 1998, vol. 8, pp. 1202–1215.
34. Marsan, L. and Sagot, M.-F., *Proc. 4th Annu. Int. Conf. Computational Molecular Biology RECOMB'2000*, 2000, pp. 210–219.
35. Ulyanov, A.V. and Stormo, G.D., *Nucleic Acids Res.*, 1995, vol. 23, pp. 1434–1440.
36. Fraenkel, Y.M., Mandel, Y., Friedberg, D., and Margalit, H., *Comput. Appl. Biosci.*, 1995, vol. 11, pp. 379–387.
37. Rocke, E. and Tompa, M., *Proc. 2nd Annu. Int. Conf. on Computational Molecular Biology RECOMB'98*, 1998, pp. 228–233.
38. Tompa, M., *Proc. 7th Int. Conf. on Intelligent Systems for Molecular Biology ISMB'1999*, 1999, pp. 262–271.
39. Rigoutsos, I. and Floratos, A., *Bioinformatics*, 1998, vol. 14, pp. 55–67.
40. van Helden, J., Andre, B., and Collado-Vides, J., *J. Mol. Biol.*, 1998, vol. 281, pp. 827–842.
41. Jensen, L.J., Knudsen, S., *Bioinformatics*, 2000, vol. 16, pp. 326–333.
42. Cho, R.J., Campbell, M.J., Winzeler, E.A., *et al.*, *Molecular Cell*, 1998, vol. 2, pp. 65–73.
43. Wolfsberg, T.G., Gabrielian, A.E., Campbell, M.J., *et al.*, *Genome Res.*, 1999, vol. 9, pp. 775–792.
44. V'yugin, V.V., Gorbunov, K.Yu., and Lyubetskii, V.A., *Trudy konferentsii "Problemy upravleniya i modelirovaniya v slozhnykh sistemakh"* (Proc. Conf. "Controlling and Simulation Problems in Complex Systems"), Myasnikov, V.P., Ed., Moscow: Ross. Akad. Nauk, 2000, pp. 130–137.
45. Robison, K., McGuire, A.M., and Church, G.M., *J. Mol. Biol.*, 1998, vol. 284, pp. 241–254.
46. Mironov, A.A., Vinokurova, N.P., and Gel'fand, M.S., *Mol. Biol.*, 2000, vol. 34, pp. 253–264.
47. Maas, W.K., *Microbiol. Rev.*, 1994, vol. 58, pp. 631–640.
48. Busby, S. and Kolb, A., *Regulation of Gene Expression in Escherichia coli*, Lin, E.C.C. and Lynch, A.S., Eds., E.G. Landes, 1995, ch. 12, pp. 255–279.