УДК 004.056.55

Ю. О. Чернышев, А. С. Сергеев, А. Н. Рязанов
Донской государственный технический университет

Оценка эффективности и сравнительные характеристики биоинспирированных алгоритмов криптоанализа

Рассматривается задача криптоанализа с использованием новой модели оптимизационных стратегий – биоинспирированных алгоритмов. Статья посвящена актуальной проблеме – определению эффективности и сравнительных характеристик биоинспирированных алгоритмов, получающих в последнее время все более широкое применение для решения широкого круга комбинаторных оптимизационных научно-технических задач, в том числе задач криптоанализа. Рассматриваются генетический алгоритм, а также алгоритм муравьиных колоний, их применение для реализации шифров перестановок и замены. Приводится псевдокод, представляющий реализацию основных операций методов криптоанализа, а также расчетные формулы, определяющие трудоемкость алгоритма. Отмечен также основной вывод, что эффективность биоинспирированных методов криптоанализа по сравнению с классическими методами в общем случае возрастает с увеличением размерности задачи.

Криптоанализ, биоинспирированные алгоритмы, генетический алгоритм, алгоритм муравьиных колоний, трудоемкость алгоритма, эффективность, шифр перестановок

В настоящее время научное направление «природные вычисления», объединяющее математические методы, в которых заложен принцип природных механизмов принятия решений, получает все более широкое распространение для решения различного круга оптимизационных задач, в том числе задач криптоанализа. В данных методах и моделях основным определяющим элементом является построение начальной модели и правил, по которым она может изменяться (эволюционировать). В течение последних лет были предложены разнообразные схемы эволюционных вычислений: генетический алгоритм (ГА), генетическое программирование, эволюционные стратегии, эволюционное программирование, модели поведения роя пчел, стаи птиц и колонии муравьев, модели отжига и другие конкурирующие эвристические алгоритмы [1]. В [2] авторами рассматривалось решение задач криптоанализа, относящихся к переборным задачам с экспоненциальной временной сложностью: традиционных симметричных криптосистем, использующих шифры перестановки и замены, а также шифров гаммирования с применением генетических алгоритмов, в [3] – симметричных и асимметричных криптосистем с использованием биоинспирированных алгоритмов муравьиных и пчелиных колоний. В [4]–[6] исследована возможность применения методов генетического поиска для реализации криптоанализа блочных криптосистем. Новые модели биоинспирированных методов глобальной оптимизации, такие, как методы, имитирующие поведение лягушек, кукушек, светлячков, и метод, имитирующий распространение сорняков, проанализированы в [7]. Основная особенность этих методов состоит в возможности поиска глобального экстремума многоэкстремальных целевых функций с большим числом переменных. Вопросы разработки и реализации современных стохастических популяционных алгоритмов решения однокритериальной задачи оптимизации рассмотрены в [8], где также предложены алгоритмы повышения эффективности этих алгоритмов посредством их гибридизации. Примеры комбинированных алгоритмов на основе высокоуровневой гибридизации вложением приведены в [9], [10].

Тем не менее, следует заметить, что, несмотря на все возрастающее применение биоинспирированных методов и алгоритмов, имитирующих процессы эволюции живой природы, при решении широкого круга оптимизационных задач (в том числе задач криптоанализа), в отечественных и зарубежных публикациях (в том числе в Интернете) не приводится каких-либо реальных оценок их трудоемкости, эффективности и сравнительных характеристик. В данной статье рассматриваются методы криптоанализа, разработанные и описанные авторами в [2]–[4], и оценивается их

эффективность и трудоемкость по сравнению с классическими методами перебора.

Генетические алгоритмы криптоанализа шифров перестановок. Среди используемых классических методов шифрования перестановками, описанных в [2], можно отметить применение шифрующих таблиц, а также маршрутные перестановки. К шифрам перестановок относятся методы шифрования, преобразования из которого изменяют только порядок следования символов исходного текста, но не изменяют их самих. Как отмечено в [11], преобразование, осуществляемое шифром перестановок и предназначенное для шифрования сообщения длиной n символов, можно представить с помощью таблицы

$$\begin{pmatrix} 1 & 2 \dots n \\ i_1 & i_2 \dots i_n \end{pmatrix},$$

где i_1 – номер места шифртекста, на которое попадает первая буква исходного сообщения при выбранном преобразовании; i_2 – номер места для второй буквы и т. д. В верхней строке таблицы выписаны по порядку числа от 1 до n, а в нижней – те же числа, но в произвольном порядке. Такая таблица называется подстановкой степени n. Отсюда следует, что число различных преобразований шифра перестановки, предназначенного для шифрования сообщений длины n, меньше либо равно $n!$.

Следовательно, несмотря на многообразие основных методов шифрования перестановками, описанных, например, в [12] (шифр «скитала»; простые шифрующие таблицы; шифрующие таблицы с одиночной перестановкой по ключу; шифрующие таблицы с двойной перестановкой по ключу; магические квадраты), основная идея криптоанализа данных шифров заключается в определении оптимального варианта подстановки, определяющего искомый текст.

Наряду с известными классическими вероятностными методами случайного поиска решения задачи криптоанализа (использование статистических свойств текста) в настоящее время получают все более широкое распространение методы направленно-случайного поиска, в том числе методы, основанные на моделировании процессов эволюции живой природы (биоинспирированные методы и алгоритмы). Наиболее известными представителями данных методов являются ГА, применение которых к задачам криптоанализа, как отмечено в [2], [13], является новым и недостаточно изученным направлением. Разработка и описание ГА для криптоанализа шифров перестановок, а также экспериментальные результаты приведены авторами в [2]. Структурную схему ГА криптоанализа можно представить в виде рис. 1.

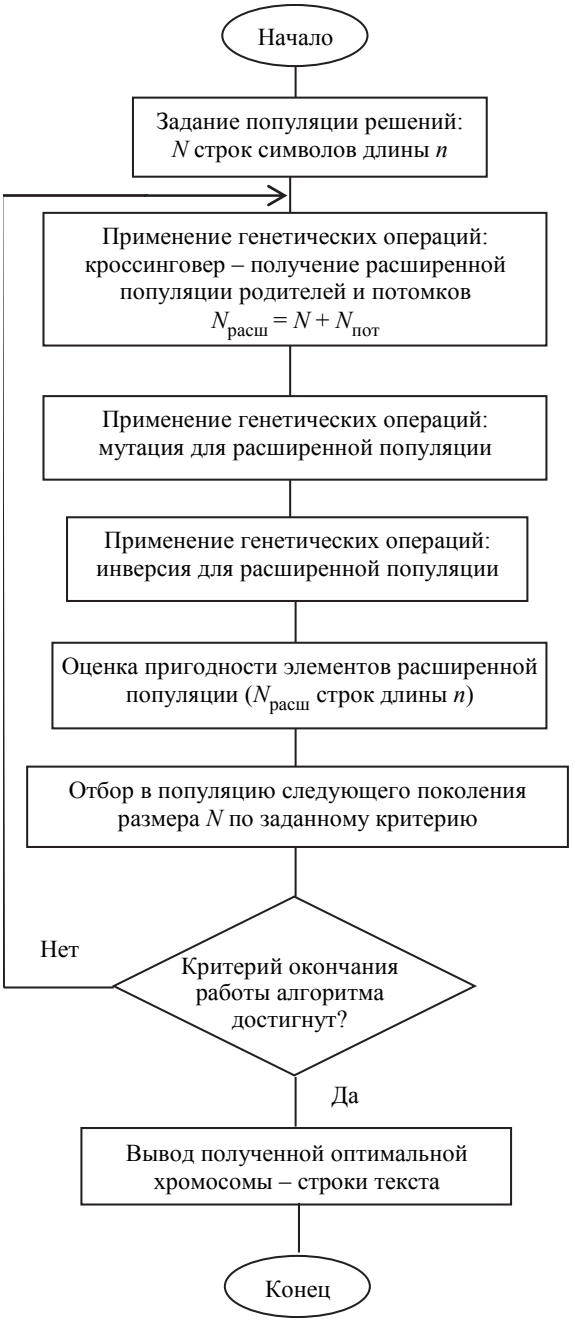

Рис. 1

Для получения оценок трудоемкости воспользуемся основными правилами оценки, описанными, например, в [14], [15]. Представим структурную схему алгоритма (рис. 1) в виде псевдокода (табл. 1), в котором N – размер популяции, n – длина хромосомы популяции, равная длине фрагмента текста.

Таблица 1

№	Оператор
	/*Формирование популяции решений*/
1	For i = 1 *to N*
2	*For j* = 1 *to n*
3	*Pop(i, j) = random (алфавит)*
4	*Next j*
5	*Next i*
	/*Наращивание счетчика поколений*/
6	*IT: Iter=Iter* + 1
	/*Применение операции кроссинговера*/
7	*For k=N* + 1 *to* $N_{\text{расш}}$
	/*Случайный выбор пары родительских хромосом*/
8	*I = random(N)*
9	*J = random(N)*
	/*Формирование потомка с помощью универсального кроссинговера*/
10	*T = random*(2)
11	*For m* =1 *to n*
12	*If T == 1 then Pop(k,m) = = Pop(i,m) else Pop(k,m) = Pop(j,m)*
13	*Next m*
14	*Next k*
	/*Проведение мутации посредством случайной замены случайного гена в случайной хромосоме */
15	*For i* = 1 *to* $N_{\text{расш}}$
16	*T = random*(100)
17	*If T <=* $100 \cdot n_{\text{мут}}$ *then j = random(n); Pop(i,j) = random* (*алфавит*)
18	*Next i*
	/*Оценка функции пригодности элементов расширенной популяции посредством сравнения каждого символа хромосом с соответствующим символом исходного текста*/
19	*For i = 1 to* $N_{\text{расш}}$
20	*S(i) = 0*
21	*Next i*
22	*For i* = 1 *to* $N_{\text{расш}}$
23	*For j* = 1 *to n*
24	*If Pop(i,j) == исходный_текст then S(i) = S(i)* + 1
25	*Next j*
26	*Next i*
	/*Проведение элитного отбора в новую популяцию размера N*/
27	*For i* =1 *to N*
28	$S_{\max} = 0$
29	*For j*=1 *to* $N_{\text{расш}}$
30	*If (S(j) >* $S_{\max}$ *then* $S_{\max}$*=S(j); t = j*
	/*Вывод оптимальной хромосомы, если она полностью совпадает с исходным текстом, и конец работы алгоритма*/
31	*If S(j) == n then*: *for k* = 1 *to n*; *print Pop(t,k)*; *Next k; Stop*
32	*Next j*
33	*For j* = 1 *to n*
34	*Pop(i,j) = Pop(t,j)*
35	*Next j*
36	*S(t)=0*
37	*Next i*
	/*Проверка достижения заданного числа поколений, если оно достигнуто, вывод популяции Pop и конец работы алгоритма*/
38	*If Iter == Iterzad then*: *for i* = 1 *to N; for j* = 1 *to n*; *print Pop(i,j)*; *Next j*; *Next i*; *Stop*, *else go to IT*

Для определения трудоемкости алгоритма воспользуемся основными правилами и формулой, представленными в [14], [15]:

$$F_{\text{цикл}} = 1 + 3N + Nf_{\text{тело цикла}}. \qquad (1)$$

Следует заметить, что алгоритм выполняет одинаковое количество операций при фиксированных входных параметрах N, n и, следовательно, является *количественно-зависимым* [14], [15].

Применяя методику анализа цикла с операторами 1–5, в соответствии с (1) получим: $F_{\text{внутр. цикл 2–4}} = 1 + 3n + n \cdot 2$ (в теле цикла содержится 2 операции: присваивание и функция «random»);

$$F_{\text{внешн. цикл 1–5}} = 1 + 3N + N \cdot F_{\text{внутр. цикл 2–4}}.$$

Тогда в соответствии с (1) получим:

$$F_{\text{цикл 1–5}} = 1 + 3N + N(1 + 3n + n \cdot 2) = \\ = 1 + 4N + 5Nn = O(Nn).$$

Оператор 6 содержит 2 элементарных операции: присваивание и сложение, его трудоемкость $F_6 = O(2)$.

В операторах 7–14 содержатся 2 вложенных цикла:

$$F_{\text{внутр. цикл 11–13}} = 1 + 3n + n\left(0.5 \cdot 1 + 0.5 \cdot 1\right) = \\ = 1 + 3n + n = 1 + 4n$$ (в операторе 12 в блоках «then» и «else» содержится по одному оператору, каждый выполняется с вероятностью 0.5). В этом случае (так как в теле цикла есть операторы 8, 9, 10 с трудоемкостью 2)

$$F_{\text{внешн. цикл 7–14}} = 1 + 3\left(N_{\text{расш}} - N\right) + \\ + \left(N_{\text{расш}} - N\right) F_{\text{внутр. цикл 11–13}} = 1 + 3\left(N_{\text{расш}} - N\right) + \\ + \left(N_{\text{расш}} - N\right)(6 + 1 + 4n) = 1 + 10N_{\text{расш}} - \\ - 10N + 4nN_{\text{расш}} - 4nN = O(nN_{\text{расш}}).$$

Аналогично, трудоемкость цикла, включающего операторы 15–18, составит (оператор 16 имеет трудоемкость 2, оператор 17 выполняется с вероятностью, равной норме мутации $n_{\text{мут}}$):

$$F_{\text{цикл 15–18}} = 1 + 3N_{\text{расш}} + N_{\text{расш}}(2 + n_{\text{мут}}(2 + 2)) = \\ = 1 + 3N_{\text{расш}} + N_{\text{расш}}(4n_{\text{мут}} + 2) = O(N_{\text{расш}} n_{\text{мут}}).$$

Трудоемкость цикла $F_{\text{цикл 19–21}} = 1 + 3N_{\text{расш}} + \\ + N_{\text{расш}} \cdot 1 = 1 + 4N_{\text{расш}}$.

Оценим трудоемкость вложенных циклов $F_{\text{цикл }22–26}$. Если использовать в качестве оценки вероятности оптимальности хромосомы (т. е. то, что символ хромосомы совпадет с символом исходного текста) значение функции Якобсена $F_{\text{якоб}}$, описанной, например, в [2], [13], то трудоемкость цикла $F_{\text{цикл }23–25} = 1 + 3n + nF_{\text{якоб}} \cdot 2$ (оператор в блоке «Then» содержит 2 операции: присваивание и сложение). Таким образом,

$$F_{\text{цикл}22–26} = 1 + 3N_{\text{расш}} + N_{\text{расш}} F_{\text{цикл}23–25} = \\ = 1 + 3N_{\text{расш}} + N_{\text{расш}}(1 + 3n + nF_{\text{якоб}} \cdot 2) = \\ = O(nN_{\text{расш}}).$$

Трудоемкость функции Якобсена может быть оценена как просмотр $n - 1$ биграмм в строке текста из n символов и подсчет количества каждой из них, а также как подсчет разности между количеством каждой из них и среднестатистической частотой встречаемости. Поэтому в качестве оценки трудоемкости можно принять $F_{\text{якоб}} = O(n)$.

Для подсчета трудоемкости цикла $F_{\text{цикл}27–37}$ определим трудоемкость вложенных циклов. В наихудшем случае, когда массив S идет по возрастанию, трудоемкость оператора 30, выполняемого с вероятностью 1, составит $F_{30} = 2$ (вследствие наличия двух операторов присваивания), трудоемкость цикла в операторе 31 $F_{\text{цикл}31} = 1 + 3n + n \cdot 1 = 1 + 4n$. Вероятность выполнения оператора 31 (вероятность, что хромосома содержит оптимальный текст) примем равной $F_{\text{якоб}}$. В этом случае $F_{31} = F_{\text{якоб}}(1 + 4n + 1)$ (вследствие наличия оператора «stop»). Таким образом, $F_{\text{цикл }29–32} = 1 + 3N_{\text{расш}} + N_{\text{расш}}(2 + \\ + F_{\text{якоб}}(1 + 4n + 1))$. Аналогично, $F_{\text{цикл}33–35} = 1 + \\ + 3n + n \cdot 1 = 1 + 4n$ (в теле цикла один оператор присваивания). Отсюда трудоемкость всего цикла

$$F_{\text{цикл}27–37} = 1 + 3N + N(1 + 1 + 3N_{\text{расш}} + \\ + N_{\text{расш}}(2 + F_{\text{якоб}}(1 + 4n + 1)) + 1 + 4n + 1) = \\ = 1 + 7N + 5NN_{\text{расш}} + 4Nn + 2F_{\text{якоб}} NN_{\text{расш}} + \\ + 4F_{\text{якоб}} NN_{\text{расш}} n = O(NN_{\text{расш}} n).$$

Трудоемкость выполнения циклов, осуществляющих вывод популяции, в операторе 38 определим как $F_{\text{цикл}38} = 1 + 4N + 4Nn + 1$ (вследствие наличия оператора «stop»). Вероятность выполнения этого оператора определяется количеством итераций «*Iterzad*», вероятность блока «then» – 1/*Iterzad*, блока «else» – (*Iterzad* – 1)/*Iterzad* и составит

$$F_{38} = (1/Iterzad)(2 + 4N + 4Nn) + \\ + \big((Iterzad - 1)/Iterzad\big) \cdot 1 = \\ = \big(1 + 4N + 4Nn + Iterzad\big)/Iterzad.$$

Таким образом, общая трудоемкость выполнения одной итерации ГА составит:

$$F_{1–38} = 1 + 4N + 5Nn + /*операторы\ 1–5*/ \\ + 2 + 1 + 10N_{\text{расш}} - 10N + 4nN_{\text{расш}} - 4nN + \\ /*операторы\ 6–14*/ \\ + 1 + 3N_{\text{расш}} + 4N_{\text{расш}} n_{\text{мут}} + 2N_{\text{расш}} + \\ /*операторы\ 15–18*/ \\ + 1 + 4N_{\text{расш}} + /*операторы\ 19–21*/ \\ + 1 + 4N_{\text{расш}} + 3N_{\text{расш}} n + 2N_{\text{расш}} nF_{\text{якоб}} + \\ /*операторы\ 22–26*/ \\ + 1 + 7N + 5NN_{\text{расш}} + 4Nn + 2F_{\text{якоб}} NN_{\text{расш}} + \\ + 4F_{\text{якоб}} NN_{\text{расш}} n + \\ /*операторы\ 27–37*/ \\ + \big(1 + 4N + 4Nn + Iterzad\big)/Iterzad \\ /*оператор\ 38*/.$$

Преобразуя данное выражение, получим, что

$$F_{1–38} = 8 + N + 23N_{\text{расш}} + 5Nn + 7N_{\text{расш}} n + \\ + 5NN_{\text{расш}} + 4N_{\text{расш}} n_{\text{мут}} + \\ + 2N_{\text{расш}} F_{\text{якоб}}(n + N + 2Nn) + \\ + (1 + 4N + 4Nn + Iterzad)/Iterzad. \qquad (2)$$

Таким образом, как следует из (2), трудоемкость итерации ГА определяется в общем случае размером популяции N и расширенной популяции $N_{\text{расш}}$, а также размером фрагмента текста n и может быть представлена как $F_{1–38} = O(NN_{\text{расш}} n)$. Отсюда следует, что в общем случае трудоемкость реализации ГА удовлетворяет условию $F_{\text{ГА}} \le Iterzad \cdot F_{1–38}$, т. е. не превосходит произведения числа итераций на трудоемкость одной итерации $F_{1–38}$, в то время как трудоемкость алгоритмов криптоанализа, основанных на технологии полного перебора, составляет $n!$. Из этого следует, что эффективность применения ГА по сравнению с классическими методами криптоанализа в общем случае возрастает с увеличением размерности задачи (фрагмента текста).

В плане сравнения эффективности данного метода с известными методами оценим трудоемкость ГА, разработанного в [13] (несмотря на то,

что авторами не приводятся сравнительные оценки разработанного метода). В указанной работе предлагается использовать числовые хромосомы, в которых отдельными генами являются элементы подстановки, т. е. i-м геном особи P является число p_i. Оператор кроссинговера, предлагаемый в [13], заключается в выборе очередного гена от одного из родителей при наличии возможности, если гены в обоих родителях недопустимы, брать произвольное допустимое значение. Его трудоемкость составляет $O(n)$, где n – длина блока текста. В качестве операции мутации используется стандартный оператор обмена двух генов хромосомы, его вычислительная сложность также $O(n)$. Условием окончания для данного ГА является превышение количества заданного числа поколений M. Следовательно, с учетом формирования начальной популяции решений трудоемкость ГА составит $O(NnM)$, где N – размер популяции; n – размер блока текста; M – количество итераций (поколений). Отсюда вытекает, что, так как функция Якобсена всегда удовлетворяет условию $F_{якоб} \neq 0$ и ГА, предлагаемый в [13], всегда проходит заданное количество поколений M, и при этом получаемый в результате дешифровки текст далеко не всегда совпадает с исходным открытым текстом, в ряде случаев ГА, предлагаемый в данной работе, может обеспечить более быструю сходимость при числе итераций, меньшем заданного числа M.

Криптоанализ шифров замены. Применительно к реализации криптоанализа шифров замены с помощью ГА отметим, что в этом случае основная цель ГА заключается в определении оптимального ключа, который должен обеспечить получение оптимального исходного текста. В [2] описаны результаты эксперимента для ряда поточных шифров (аффинный шифр и система Цезаря), а также блочных шифров замены (шифры Плейфейра и Уитсона) по определению ключевого слова. Заметим, что, как отмечено в [4], отличительной особенностью применения биоинспирированных методов криптоанализа (в частности, ГА) является возможность использования самого алгоритма шифрования (или расшифровки) в качестве целевой функции для оценки пригодности ключа, определенного с помощью генетических операций. Поэтому можно утверждать, что при использовании ГА процесс определения секретного ключа (например, при криптоанализе 2-го типа) зависит не столько от сложности шифрующих преобразований, сколько от самого биоинспирированного метода, который должен обеспечивать достаточное разнообразие генерации ключей. Структурную схему криптоанализа шифров замены с помощью ГА можно представить в виде рис. 2.

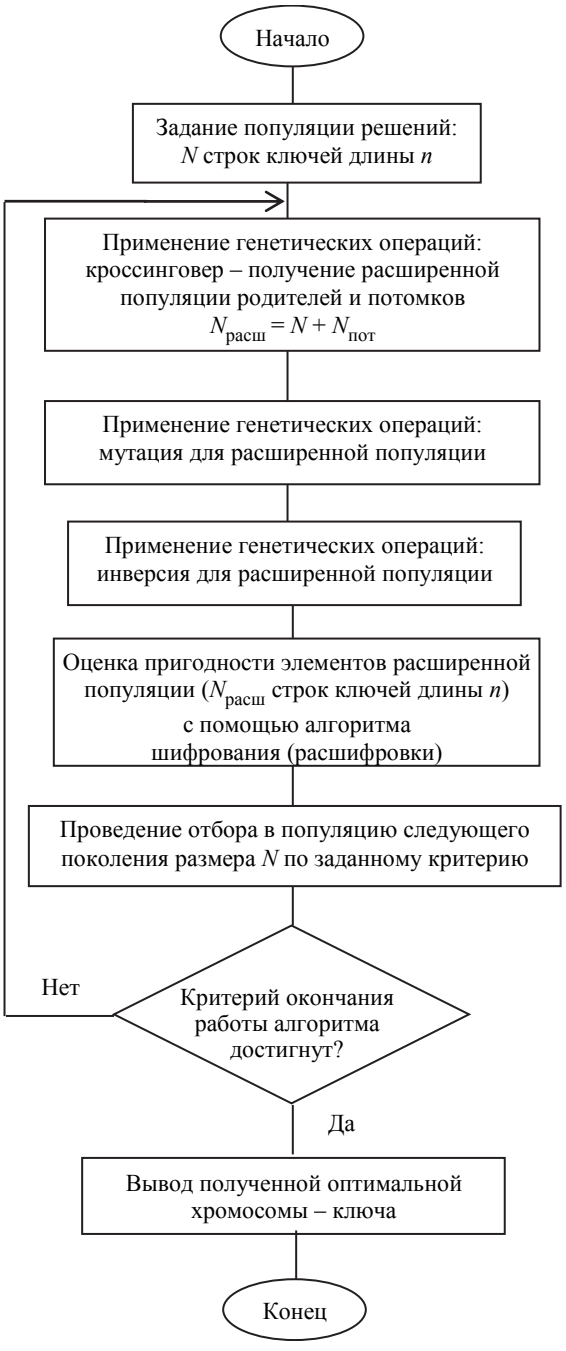

Рис. 2

Из данной схемы следует, что трудоемкость ГА криптоанализа определяется трудоемкостью самого алгоритма шифрования $F_{шифр}$. Псевдокод алгоритма, аналогично приведенному выше, представим в виде табл. 2 (как и ранее, N – размер популяции; n – длина хромосомы популяции (длина ключа); $N_{текст}$ – длина блока текста).

Таблица 2

№	Оператор
/*Формирование популяции решений*/	
1	For i = 1 *to N*
2	*For j* = 1 *to n*
3	*Pop(i, j) = random (алфавит_ключа))*
4	*Next j*
5	*Next i*
/*Наращивание счетчика поколений*/	
6	*IT: Iter=Iter* + 1
/*Применение операции кроссинговера*/	
7	*For k=N* + 1 *to* $N_{\text{расш}}$
/*Случайный выбор пары родительских хромосом*/	
8	*I = random(N)*
9	*j = random(N)*
/*Формирование потомка с помощью универсального кроссинговера*/	
10	*T = random*(2)
11	*For m* =1 *to n*
12	*If T == 1 then Pop(k,m) = = Pop(i,m) else Pop(k,m) = Pop(j,m)*
13	*Next m*
14	*Next k*
/*Проведение мутации посредством случайной замены случайного гена в случайной хромосоме */	
15	*For i* = 1 *to* $N_{\text{расш}}$
16	*T = random*(100)
17	*If T <=* 100·$n_{\text{мут}}$ *then j = random(n); Pop(i,j) = random* (*алфавит*)
18	*Next i*
/*Оценка функции пригодности элементов расширенной популяции посредством реализации алгоритма шифрования, использующего ключ – элемент популяции*/	
19	*For i = 1 to* $N_{\text{расш}}$
20	*For j = 1 to n*
21	*S(i) =* $F_{\text{шифр}}$*(Pop(i,j))*
22	*Next j*
23	*Next i*
/*Проведение элитного отбора в новую популяцию размера N*/	
24	*For i* =1 *to N*
25	$S_{\max}$ = 0
26	*For j*=1 *to* $N_{\text{расш}}$
27	*If (S(j) >* $S_{\max}$ *then* $S_{\max}$*=S(j); t = j*
/*Вывод оптимальной хромосомы, если она полностью совпадает с исходным текстом, и конец работы алгоритма*/	
28	*If S(j) ==* $N_{\text{текст}}$ *then*: *for k* = 1 *to n*; *print Pop(t,k)*; *Next k; Stop*
29	*Next j*
30	*For j* = 1 *to n*
31	*Pop(i,j) = Pop(t,j)*
32	*Next j*
33	*S(t)=0*
34	*Next i*
/*Проверка достижения заданного числа поколений, если оно достигнуто, вывод популяции Pop и конец работы алгоритма*/	
35	*If Iter == Iterzad then*: *for i* = 1 *to N; for j* = 1 *to n*; *print Pop(i,j)*; *Next j*; *Next i*; *Stop*, *else go to IT*

Таким образом, отличительными особенностями применения ГА для криптоанализа шифров замены является формирование начальной популяции решений из символов алфавита ключа (вариантов ключа), а также оценка функции пригодности ключа $F_{\text{шифр}}$ с использованием алгоритма шифрования. В данном случае совпадение полученного текста с исходным определяется функцией $F_{\text{шифр}}$. Если число совпадений полученного при использовании ключа i текста равно длине исходного текста $N_{\text{текст}}$, то ключ является оптимальным и осуществляется вывод данного элемента популяции.

Трудоемкость данного ГА, как видно из приведенного фрагмента псевдокода, определяется функцией пригодности ключа – трудоемкостью шифрующего алгоритма. Как и ранее, $F_{\text{цикл1–5}} = 1 + 4N + 5Nn = O(Nn)$.

Трудоемкость цикла $F_{\text{цикл 19–23}}$, осуществляющего оценку функции пригодности ключей, примем равной $F_{\text{цикл 19–23}} = 1 + 3N_{\text{расш}} + N_{\text{расш}} \times$ $\times (1 + 3n + n(F_{\text{шифр}} + 1))$, так как оператор 21 содержит функцию $F_{\text{шифр}}$, а также оператор присваивания. Если принять вероятность получения оптимальной хромосомы (оптимального ключа) на каждой итерации $P_{\text{ГА}}$ (в общем случае можно считать, что она определяется параметрами задачи, настройкой параметров ГА и частотными характеристиками текста, в качестве нее можно принять значение функции Якобсена $F_{\text{якоб}}$), то $F_{28} = P_{\text{ГА}}(1 + 4n + 1)$. Поэтому общую трудоемкость одной итерации ГА криптоанализа шифра замены можно представить следующим выражением:

$$F_{1–35} = 1 + 4N + 5Nn + \text{/*операторы 1–5*/}$$
$$+ 2 + 1 + 10N_{\text{расш}} - 10N + 4nN_{\text{расш}} - 4nN +$$
$$\text{/*операторы 6,7–14*/}$$
$$+ 1 + 3N_{\text{расш}} + 4N_{\text{расш}}n_{\text{мут}} + 2N_{\text{расш}} +$$
$$\text{/*операторы 15-18*/}$$
$$+ 1 + 4N_{\text{расш}} + 4N_{\text{расш}}n + F_{\text{шифр}}N_{\text{расш}}n +$$
$$\text{/*операторы 19–23*/}$$
$$+ 1 + 7N + 5NN_{\text{расш}} + 4Nn + 2P_{\text{ГА}}NN_{\text{расш}} +$$
$$+ 4P_{\text{ГА}}NN_{\text{расш}}n + \text{ /*операторы 24–34*/}$$
$$+ \left(1 + 4N + 4Nn + Iterzad\right) / Iterzad \text{ /*оператор 35*/.}$$

Преобразуя выражение, получим

$$F_{1-35} = 7 + N + 19N_{\text{расш}} + 5Nn + 8N_{\text{расш}}n + \\ + 5NN_{\text{расш}} + 4N_{\text{расш}}n_{\text{мут}} + F_{\text{шифр}}N_{\text{расш}}n + \\ + 2P_{\text{ГА}}NN_{\text{расш}}(1+2n) + \\ + (1 + 4N + 4Nn + Iterzad)/Iterzad. \quad (3)$$

Таким образом, как следует из (3), трудоемкость ГА в данном случае определяется размерами исходной и расширенной популяций N и $N_{\text{расш}}$, подбором параметров ГА, обеспечивающих вероятность получения оптимальной хромосомы-ключа $P_{\text{ГА}}$ и определяющих число итераций для получения результата, а также размером ключа n и трудоемкостью используемого алгоритма шифрования $F_{\text{шифр}}$. Результаты эксперимента для ряда шифров замены при известной и неизвестной длине ключевого слова приведены в [2]. Как показывает (3), при увеличении длины ключа n и линейной или степенной трудоемкости алгоритма шифрования $F_{\text{шифр}}$ эффективность работы ГА очевидным образом повышается по сравнению с факториальной сложностью $n!$ определения ключа методом перебора.

Среди известных методов криптоанализа шифров замены с помощью биоинспирированных методов отметим работу [16]. В ней приводится описание оригинального ГА криптоанализа шифра Вижинера (шифра многоалфавитной замены) без оценки вычислительной сложности и трудоемкости. В данном алгоритме кодирование хромосом, представляющих популяцию ключей, осуществляется присвоением генам числовых значений, соответствующих порядковым номерам символов в алфавите. Мутация хромосомы заключается в произвольном обмене значений двух генов в хромосоме с заданной вероятностью p_0, используется одноточечный кроссинговер, особи для которого выбираются с помощью «колеса рулетки». В качестве целевой функции пригодности используется функция Якобсена [13]. Алгоритм реализуется в 2 этапа: на 1-м этапе определяется длина секретного ключа; на 2-м – сам ключ (аналогично [2]). Если длина ключевого слова известна, можно считать, что итерация описанного в [16] ГА имеет трудоемкость реализации, сопоставимую с итерацией ГА, описанного выше.

Криптоанализ шифров перестановок с использованием алгоритма муравьиных колоний. В настоящее время ГА в основном используются для поиска оптимальных и квазиоптимальных решений в таких задачах, в которых оптимальное решение может быть получено только полным перебором за экспоненциальное время. Однако недостатком ГА является наличие «слепого» поиска, что в общем случае приводит к таким отрицательным моментам, как генерация решений с нарушениями (что приводит к увеличению времени поиска и требует дополнительного контроля); генерация большого количества одинаковых решений; генерация большого количества плохо приспособленных решений (что в общем случае может привести к попаданию в локальный оптимум) [3]. В связи с этим представляет интерес применение эвристических подходов, инспирированных природными системами, в которых осуществляется поэтапное построение решения задачи (т. е. добавление нового оптимального частичного решения к уже построенному частичному оптимальному решению).

Возможный подход к реализации криптоанализа шифров перестановки на основе сведения данной проблемы к классической задаче о назначениях, решаемой с помощью алгоритма муравьиных колоний, описан в [3], [17]. Несмотря на широкое применение муравьиных алгоритмов для решения комбинаторных оптимизационных задач, представимых в виде взвешенных графовых моделей и связанных с определением оптимальных маршрутов, какие-либо оценки трудоемкости и сложности методов данного класса и их сравнительные характеристики в отечественных и зарубежных публикациях, а также в Интернете обнаружить не удалось. Возможно, это связано со случайным характером процесса поиска в биоинспирированных алгоритмах, обеспечивающего равновероятный поиск оптимального элемента равнозначно на любой итерации (т. е. количество итераций и трудоемкость поиска являются, по сути, случайными величинами).

Алгоритм криптоанализа шифров перестановок, приведенный в [3], [17], представим в виде следующего псевдокода: C – матрица вероятности соседства символов; Fer – матрица концентраций феромона; ρ – коэффициент испарения; m – количество маршрутов муравьев; n – длина маршрута. Данный алгоритм также является *количественно-зависимым* по трудоемкости, так как его функция трудоемкости зависит от размерности входных данных, а не от их конкретных значений (табл. 3).

Таблица 3

№	Оператор
/*Формирование *m* маршрутов муравьев длины *n**/	
1	For *k* = 1 *to m*
2	*For i* = 1 *to n*
3	*Mar(k,i) = random (алфавит)*
4	*Next i*
5	*Next k*
/*Подсчет критериев оптимальности маршрутов*/	
6	For *k* = 1 *to m*
7	*S(k)* = 0
8	*For i* = 1 *to n*–1
9	*S(k) = S(k) + C(Mar(k,i), Mar(k,i*+1*)*
10	*Next i*
11	*Next k*
/*Умножение функции качественности на весовой коэффициент*/	
12	For *k* = 1 *to m*
13	*For i* = 1 *to n*
14	$T = F_{якоб}(Mar(k,i))$
15	*S(k) = S(k)·T*
16	*Next i*
17	*Next k*
/*Наращивание счетчика поколений*/	
18	*IT:Iter = Iter* + 1
/*Подсчет матрицы результирующих концентраций феромона*/	
19	*For l* = 1 *to n*
20	*V = алфавит(l)*
21	*W = 0*
22	*For i*=1 *to n*
23	*For k* = 1 *to m*
24	*If(Mar(k,i) == v) then w = w + S(k)*
25	*Next k*
26	*Fer(l,i) = w*
27	*Next i*
28	*Next l*
/*Проведение испарения феромона*/	
29	*For i*=1 *to n*
30	*For j*=1 *to n*
31	*Fer(i,j) = Fer(i,j)*(1 – ρ)
32	*Next j*
33	*Next i*
/*Вычисление вероятностей размещения символов в позиции*/	
34	*For i*=1 *to n*
35	*W = 0*
36	*For k* = 1 *to n*
37	*W = w + Fer(k,i)*
38	*Next k*
39	*For k* = 1 *to n*
40	*Fer(k,i) = Fer(k,i)/w*
41	*Next k*
42	*Next i*
/*Формирование новых *dm* маршрутов*/	
43	*For j=m*+1 *to m + dm*
44	*For l* = 1 *to n*
45	*W = random*(1)
46	*V = random(n)*
47	*If(Fer(v,l) <= w) then Mar(j,l) = алфавит(Fer(v,l));l = n*
48	*Next l*
49	*Next j*

Окончание табл. 3

№	Оператор
/*Подсчет критериев оптимальности новых *dm* маршрутов*/	
50	*For k = m* + 1 *to m + md*
51	*S(k)* = 0
52	*For i* = 1 *to n*–1
53	*S(k) = S(k) + C(Mar(k,i),Mar(k,I* + 1*)*
54	*Next i*
55	*Next k*
/*Умножение функции качественности на весовой коэффициент для новых *dm* маршрутов*/	
56	*For k = m* + 1 *to m + md*
57	*For i*=1 *to n*
58	$T = F_{якоб}(Mar(k,i))$
59	*S(k) = S(k)·T*
60	*Next i*
61	*Next k*
/*Выбор *m* маршрутов с лучшими значениями функции качественности*/	
62	*For i*=1 *to m*
63	$S_{max} = 0$
64	*For j = m* + 1 *to m + m·d*
65	*If(S(j) >* S_{max}*) then* S_{max} *= S(j); t = j*
66	*Next j*
67	*For j* = 1 *to n*
68	*Mar(i,j) = Mar(t,j)*
69	*Next j*
70	*S(t)* = 0
71	*Next i*
/*Проверка достижения заданного числа поколений, если оно достигнуто, вывод популяции маршрутов муравьев и конец работы алгоритма*/	
72	*If (Iter == Iterzad) then*: *for i* = 1 *to m; for j* = 1 *to n; print Mar(i,j); Next j; Next i; Stop else go to IT*

Таким образом, данный псевдокод имитирует реализацию муравьиного алгоритма криптоанализа шифров перестановок, описанного в [3], [17], отражает основные операции муравьиного алгоритма и позволяет оценить трудоемкость метода на основе основных правил анализа программ как сумму трудоемкостей блоков, следующих друг за другом в алгоритме [15].

В цикле операторов 1–5 осуществляется формирование множества маршрутов муравьев, в соответствии с [15] трудоемкость цикла равна $F_{\text{цикл }1\text{–}5} = 1 + 4m + 5mn = O(mn)$.

В цикле операторов 6–11 осуществляется оценка функции пригодности маршрутов, сформированных муравьями, с использованием матрицы вероятности соседства символов *C*. Трудоемкость внутреннего цикла $F_{\text{цикл }8\text{–}10} = 1 + 3(n-1) + (n-1)2$, так как в теле цикла (оператор 9) содержатся 2 арифметические операции – присваивание и сложение. В этом случае (вследствие наличия оператора 7)

$$F_{\text{цикл }6\text{–}11} = 1 + 3m + m[1 + 1 + 3(n-1) + (n-1)2] =$$
$$= 1 + 5nm = O(nm).$$

В цикле операторов 12–17 производится умножение функции пригодности маршрутов на весовой коэффициент. При использовании для этой цели функции Якобсена и организации процесса в соответствии с последовательностью операторов 12–17 трудоемкость процесса не должна превысить (вследствие наличия в операторе 14 функции Якобсена и операции присваивания, в операторе 15 – операций присваивания и умножения)

$$F_{\text{цикл }12\text{–}17} = 1 + 3m + m[1 + 3n + n(1 + F_{\text{якоб}} + 2)] = \\ = 1 + 4m + mn(6 + F_{\text{якоб}}).$$

В операторе 18 производится наращивание счетчика числа итераций, его трудоемкость (вследствие наличия двух арифметических операций: присваивания и сложения) $F_{18} = O(2)$.

В цикле, включающем операторы 19–28, формируется матрица результирующих концентраций феромона *Fer* (методика получения данной матрицы описана в [3], [17]). Если принять вероятность того, что символ маршрута муравья равен заданному символу алфавита, равной функции Якобсена, то трудоемкость внутреннего цикла (вследствие наличия оператора присваивания 26)

$$F_{\text{цикл }22\text{–}27} = 1 + 3n + n[1 + 3m + mF_{\text{якоб}}\,2 + 1] = \\ = 1 + 5n + mn(3 + 2F_{\text{якоб}}).$$

Трудоемкость внешнего цикла в этом случае $F_{\text{цикл }19\text{–}28} = 1 + 3n + n\,[1 + 1 + 1 + 5n + mn \times (3 + 2 \times \times F_{\text{якоб}})] = 1 + 6n + n^2(5 + 3m + 2mF_{\text{якоб}})$. Процедура имитации испарения феромона осуществляется в цикле, включающем операторы 29–33. В этом случае (так как оператор 31 включает 3 арифметические операции)

$$F_{\text{цикл }29\text{–}33} = 1 + 3n + n(1 + 3n + 3n) = \\ = 1 + 4n + 6n^2 = O(n^2).$$

Процедура формирования матрицы вероятностей размещения символов в позиции осуществляется в цикле, включающем операторы 34–42. Трудоемкость вложенных циклов (вследствие того, что операторы 37 и 40 содержат 2 арифметические операции)

$$F_{\text{цикл }36\text{–}38} = 1 + 3n + n \cdot 2 = 1 + 5n,$$

$$F_{\text{цикл }39\text{–}41} = 1 + 3n + 2n = 1 + 5n.$$

Таким образом,

$$F_{\text{цикл }34\text{–}42} = 1 + 3n + n[1 + (1 + 5n) + (1 + 5n)] = \\ = 1 + 6n + 10n^2 = O(n^2).$$

Формирование новых *dm* маршрутов (где *d* – априорно задаваемый параметр, $d < 1$) осуществляется в цикле, включающем операторы 43–49. Процедура, имитирующая «колесо рулетки», осуществляется следующим образом: параметру *w* присваивается произвольное значение от 0 до 1 в операторе 45 (вероятность выбора), параметру *v* – произвольное значение из мощности алфавита *n* (оператор 46). Если соответствующий элемент в матрице вероятностей размещения *Fer* в строке *v* (и столбце с номером *l*, который пробегает все значения от 1 до *n* в цикле) меньше вероятности *w*, соответствующему элементу маршрута присваивается символ алфавита, соответствующий строке *v* и столбцу *l* матрицы *Fer* (оператор 47). Следовательно, если принять в качестве вероятности выбора оптимального символа значение функции Якобсена (или ее нормированное значение), то трудоемкость внутреннего цикла (трудоемкость операторов 45 и 46 равна $O(2)$, в операторе 47 содержится 2 оператора присваивания)

$$F_{\text{цикл }44\text{–}48} = 1 + 3n + n[2 + 2 + F_{\text{якоб}} \cdot 2] = \\ = 1 + 7n + 2nF_{\text{якоб}}.$$

В этом случае

$$F_{\text{цикл }43\text{–}49} = 1 + 3(m + md - m - 1 + 1) + \\ + md(1 + 7n + 2nF_{\text{якоб}}) = \\ = 1 + 4md + nmd(7 + 2F_{\text{якоб}}).$$

В цикле, содержащем операторы 50–55, осуществляется подсчет критериев оптимальности полученных *dm* маршрутов с помощью матрицы вероятности соседства символов *C*. В этом случае трудоемкость внутреннего цикла (так как оператор 53 содержит 2 операции присваивания)

$$F_{\text{цикл }52\text{–}54} = 1 + 3(n - 1) + (n - 1)2 = 5n - 4.$$

Трудоемкость внешнего цикла (вследствие наличия оператора 51)

$$F_{\text{цикл}50\text{–}55} = 1 + 3(m + md - m - 1 + 1) + \\ + md\,(1 + 5n - 4) = 1 + 5nmd.$$

Умножение функции качественности на весовой коэффициент для новых *dm* маршрутов производится в цикле, включающем операторы 56–61. Трудоемкость вложенного цикла (вследствие наличия в операторе 58 операции присваивания и функции Якобсена, а в операторе 59 – операций умножения и присваивания)

$$F_{\text{цикл}57-60} = 1 + 3n + n(1 + F_{\text{якоб}} + 2) = \\ = 1 + 6n + nF_{\text{якоб}}.$$

Трудоемкость внешнего цикла

$$F_{\text{цикл}56-61} = 1 + 3(m + md - m - 1 + 1) + \\ + md(1 + 6n + nF_{\text{якоб}}) = 1 + 4md + nmd(6 + F_{\text{якоб}}).$$

В цикле, включающем операторы 62–71, осуществляется выбор m маршрутов с лучшими значениями функции качественности. Если массив целевых функций S отсортирован по возрастанию (худший случай), то вероятность выполнения оператора 65 равна 1 и трудоемкость вложенного цикла

$$F_{\text{цикл }64-66} = 1 + 3(m + md) + (m + md)2 = \\ = 1 + 5m + 5md.$$

Аналогично (так как оператор 68 содержит одну операцию присваивания)

$$F_{\text{цикл }67-69} = 1 + 3n + n = 1 + 4n.$$

Таким образом, трудоемкость всего цикла

$$F_{\text{цикл }62-71} = 1 + 3m + m(1 + 1 + 5m + \\ + 5md + 1 + 4n + 1) = 1 + 7m + 5m^2(1 + d) + 4mn.$$

Трудоемкость выполнения циклов в операторе 72 (вследствие наличия оператора «stop»)

$$F_{\text{цикл}72} = 1 + 3m + m(1 + 3n + n) + 1 = 2 + 4m + 4mn.$$

Отсюда, как и ранее, трудоемкость всего оператора 72

$$F_{72} = (1/Iterzad)(2 + 4m + 4mn) + \\ + \big((Iterzad - 1)/Iterzad\big) \cdot 1 = \\ = (1 + 4m + 4mn + Iterzad)/Iterzad.$$

Таким образом, общая трудоемкость выполнения одной итерации алгоритма муравьиных колоний

$$F_{1-72} = 1 + 4m + 5mn + \ /*\text{операторы } 1–5*/ \\ + 1 + 5nm + \ /*\text{операторы } 6–11*/ \\ + 1 + 4m + mn(6 + F_{\text{якоб}}) + \ /*\text{операторы } 12–17*/ \\ + 2 + 1 + 6n + n^2(5 + 3m + 2mF_{\text{якоб}}) + \\ /*\text{операторы } 18, 19–28*/ \\ + 1 + 4n + 6n^2 + \ /*\text{операторы } 29–33*/ \\ + 1 + 6n + 10n^2 + \ /*\text{операторы } 34–42*/ \\ + 1 + 4md + nmd(7 + 2F_{\text{якоб}}) + \\ /*\text{операторы } 43–49*/ \\ + 1 + 5nmd \ /*\text{операторы } 50–55*/ \\ + 1 + 4md + nmd(6 + F_{\text{якоб}}) + \\ /*\text{операторы } 56–61*/ \\ + 1 + 7m + 5m^2(1 + d) + 4mn + \\ /*\text{операторы } 62–71*/ \\ + (1 + 4m + 4mn + Iterzad)/Iterzad \\ /*\text{оператор } 72*/$$

Преобразуя данное выражение, получим, что трудоемкость одной итерации алгоритма муравьиных колоний

$$F_{1-72} = 12 + 15m + 16n + 20mn + 21n^2 + \\ + mn^2(3 + 2F_{\text{якоб}}) + 8md + nmd(18 + 3F_{\text{якоб}}) + \\ + mnF_{\text{якоб}} + 5m^2(1 + d) + \\ + (1 + 4m + 4mn + Iterzad)/Iterzad. \qquad (4)$$

Как следует из (4), трудоемкость итерации алгоритма муравьиных колоний в общем случае определяется квадратичной зависимостью от количества маршрутов и длины маршрута муравья. Таким образом, можно считать, что трудоемкость удовлетворяет условию $F_{1-72} = O(m^2 + n^2)$, при этом использовались допущения, что формирование маршрутов в цикле операторов 43–49 всегда возможно (т. е. в матрице *Fer* всегда существует элемент, удовлетворяющий условию *Fer*(v, l) <= w) и что в процессе реализации алгоритма работа завершается после просмотра заданного количества итераций *Iterzad*.

Алгоритмы муравьиных колоний могут быть эффективно использованы для решения NP-полных задач, в которых требуется определение некоторого множества элементов с заданными свойствами (в отличие от ГА, цель которого – найти глобальный оптимум на некотором множестве элементов и который в ряде случаев допускает попадание в локальный оптимум). Исследование эвристических алгоритмов муравьиных колоний показало, что применение их позволяет осуществлять поэтапное построение решения задачи, т. е. добавление нового оптимального частичного решения к уже построенному частичному оптимальному решению, что устраняет недостатки ГА «слепого» поиска (генерация решений с нарушениями, большое количество одинаковых и плохо приспособленных решений, что приводит к увеличению времени поиска и попаданию в локальный оптимум) [3], [4].

Криптоанализ шифров замены с использованием алгоритма муравьиных колоний. Применение алгоритмов муравьиных колоний для реализации криптоанализа шифров замены рассмотрено в [3], [18]. Криптоанализ шифров заме-

ны, использующих ключевые символы (цифровые или буквенные), может быть сведен к задаче определения позиций для назначения символов ключа из заданного алфавита символов таким образом, при котором целевая функция, определяющая оптимальность исходного текста, достигает экстремума. Другими словами, данная задача криптоанализа, по сути, также является частным случаем задачи о назначениях [18].

Если ключевое слово является осмысленным словом на естественном языке (т. е. его символы подчиняются среднестатистическим значениям встречаемости и сочетаемости языка) и оптимальность расшифрованного текста зависит от оптимальности определенного на данном шаге ключа, процесс криптоанализа 2-го типа (определение секретного ключа, обеспечивающего преобразование зашифрованного текста в исходный и наоборот) может быть организован аналогично описанному ранее процессу применения ГА посредством реализации самого алгоритма шифрования для оценки оптимальности определенного на каждой итерации ключа. В [3], [18] исследована зависимость получения строки исходного осмысленного текста из строки «ЯШЕОНЭОЖЙЬНИСИ» в зависимости от полученного на каждой итерации ключевого слова с оптимальным значением функции качественности: «СКОБИ», «СКОТИ», «СКАБО», «АКОБА» для системы Цезаря с ключевым словом и ключом $k = 3$. В связи с этим при реализации алгоритмов муравьиных колоний для криптоанализа шифров замены необходимо (как и при использовании ГА) введение операторов, аналогичных операторам 19–23 в псевдокоде для ГА криптоанализа шифров замены (т. е. необходимо введение оператора типа $S(i) = F_{\text{шифр}}(Mar(i,j))$.

Следовательно, в этом случае трудоемкость алгоритма криптоанализа может возрасти с учетом трудоемкости шифрующего алгоритма, используемого для оценки пригодности ключа $F_{\text{шифр}}$.

В данной статье рассмотрена актуальная задача – определение эффективности и сравнительных характеристик биоинспирированных алгоритмов для решения комбинаторных оптимизационных задач криптоанализа. Рассмотрено применение ГА и алгоритмов муравьиных колоний для реализации криптоанализа шифров перестановок и замены. На основе приведенного псевдокода, представляющего реализацию основных операций алгоритмов криптоанализа, получены расчетные формулы, определяющие трудоемкость итерации алгоритма. Приведены сравнительные характеристики представленного ГА с известными методами криптоанализа, из которых следует, что ГА, предлагаемый в [13], всегда проходит заданное количество поколений M и его трудоемкость составит $O(NnM)$ (т. е. в общем случае она определяется размером популяции N, размером блока текста n и количеством поколений M), в то время как трудоемкость представленного ГА может быть определена как $O(NN_{\text{расш}}n)$ и в общем случае удовлетворяет условию $F_{\text{ГА}} \le Iterzad \cdot F_{1-38}$, т. е. не превосходит произведения числа итераций на трудоемкость одной итерации. Отсюда следует вывод, что эффективность применения представленного ГА по сравнению с классическими методами криптоанализа в общем случае возрастает, и при этом ее оценку можно представить как $\frac{O(NnM)}{O(NN_{\text{расш}}n)}$ (т. е. эффективность возрастает с увеличением числа итераций ГА, так как в этом случае возрастает вероятность, что представленный ГА определит оптимальное решение раньше равновероятно на любой итерации).

Работа выполнена при финансовой поддержке РФФИ (проекты 17-01-00375, 18-01-00314).

СПИСОК ЛИТЕРАТУРЫ

1. Лебедев В. Б. Модели адаптивного поведения колонии пчел для решения задач на графах // Изв. ЮФУ. 2012. № 7. С. 42–49.

2. Криптографические методы и генетические алгоритмы решения задач криптоанализа / Ю. О. Чернышев, А. С. Сергеев, Е. О. Дубров, А. В. Крупенин, О. П. Третьяков. Краснодар: ФВАС, 2013. 138 с.

3. Биоинспирированные алгоритмы решения задач криптоанализа классических и асимметричных криптосистем / Ю. О. Чернышев, А. С. Сергеев, Е. О. Дубров, А. В. Крупенин, С. А. Капустин, А. Н. Рязанов. Краснодар: Изд-во КВВУ, 2015. 132 с.

4. Применение биоинспирированных методов оптимизации для реализации криптоанализа блочных методов шифрования / Ю. О. Чернышев, А. С. Сергеев, Е. О. Дубров, А. Н. Рязанов. Ростов н/Д.: Изд-во ДГТУ, 2016. 177 с.

5. Исследование возможности применения генетических алгоритмов для реализации криптоанализа

блочных криптосистем / Ю. О. Чернышев, А. С. Сергеев, Н. Н. Венцов, А. Н. Рязанов // Вестн. ДГТУ. 2015. № 3 (82). С. 65–72.

6. Исследование возможности применения методов эволюционной оптимизации для реализации криптоанализа блочных методов шифрования / Ю. О. Чернышев, А. С. Сергеев, С. А. Капустин, А. Н. Рязанов // Изв. СПбГЭТУ «ЛЭТИ». 2015. № 10. С. 32–40.

7. Орловская Н. М. Анализ эффективности биоинспирированных методов глобальной оптимизации // Тр. МАИ: электрон. науч. журн. 2014. № 73. С. 4.

8. Карпенко А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. М.: Изд-во МГТУ им. Н. Э. Баумана, 2017. 446 с.

9. Чернышев Ю. О., Сергеев А. С. Применение комбинированного биоинспирированного алгоритма (генетический алгоритм и алгоритм муравьиных колоний) для реализации криптоанализа шифров перестановок // Изв. СПбГЭТУ «ЛЭТИ». 2017. № 9. С. 33–44.

10. Чернышев Ю. О., Сергеев А. С. Применение комбинированных биоинспирированных стратегий (генетический алгоритм и алгоритм пчелиных колоний) для реализации криптоанализа классических шифров перестановок // Инженерный вестн. Дона. 2017. № 4. Электрон. журн. URL: http://ivdon.ru/ru/magazine/archive/n4y2017/4518 (дата обращения 06.04.2018).

11. Шифры перестановки. URL: http://citforum.ru/security/cryptography/yaschenko/78.html (дата обращения 06.04.2018).

12. Шифры перестановок. Шифр «скитала». URL: http://neudoff.net/info/informatika/shifry-perestanovok-shifr-skitala/ (дата обращения 06.04.2018).

13. Городилов А. Ю. Криптоанализ перестановочного шифра с помощью генетического алгоритма // Вестн. Перм. ун-та. Сер. Математика, механика, информатика. 2007. № 7. С. 44–49.

14. Трудоемкость алгоритмов и временные оценки. URL: http://th-algoritmov.narod.ru/5.htm (дата обращения 06.04.2018).

15. Анализ алгоритмов. Сравнительные оценки алгоритмов. URL: http://fsc.bsu.by/wp-content/uploads/2015/12/AiSD-Tema-Analiz-algoritmov.pdf (дата обращения 06.04.2018).

16. Морозенко В. В., Елисеев Г. О. Генетический алгоритм для криптоанализа шифра Вижинера // Вестн. Перм. ун-та. Сер. Математика, механика, информатика. 2010. № 1. С. 75–80.

17. Фатхи В. А., Сергеев А. С. Исследование возможности применения алгоритма муравьиных колоний для реализации криптоанализа шифров перестановок // Вестн. ДГТУ. 2011. Т. 11, № 1 (52). С. 10–20.

18. Исследование возможности применения биоинспирированных алгоритмов оптимизации для реализации криптоанализа шифров замены / Ю. О. Чернышев, А. С. Сергеев, Е. О. Дубров, А. Н. Рязанов // Информационное противодействие угрозам терроризма: науч.-практ. журн. 2014. № 23. С. 309–321.

Yu. O. Chernyshev, A. S. Sergeyev, A. N. Ryazanov
Don State Technical University

ASSESSMENT OF EFFICIENCY AND COMPARATIVE CHARACTERISTICS OF THE BIOINSPIRED CRYPTANALYSIS ALGORITHMS

In article the cryptanalysis task with use of new model of optimizing strategy – the bioinspired algorithms is considered. Article is devoted to a actual problem – determination of efficiency and comparative characteristics of the bioinspired algorithms which are widely used recently more and more for the solution of a wide range of combinatory optimizing scientific and technical tasks, including cryptanalysis tasks. The genetic algorithm, as well as the algorithm of ant colonies, their application to the implementation of codes of permutations and substitutions are considered. Pseudo-code representing the implementation of the basic operations of methods of cryptanalysis, as well as calculation formulas that determine the complexity of the algorithm are given. The main conclusion is also noted that the efficiency of bioinspired methods of cryptanalysis in comparison with classical methods generally increases with the increase in the dimension of the problem.

Cryptanalysis, the bioinspired algorithms, genetic algorithm, algorithm of ant colonies, algorithm complexity, efficiency, code of shifts