УДК 004.056.55

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

П. А. Панасенко
Краснодарское высшее военное училище им. генерала армии С. М. Штеменко

Оценка эффективности и сравнительные характеристики комбинированных биоинспирированных алгоритмов криптоанализа

Рассматривается решение актуальной задачи криптоанализа с использованием новой модели оптимизационных стратегий – комбинированных биоинспирированных алгоритмов. Описывается проблема определения эффективности комбинированных биоинспирированных алгоритмов, используемых в последнее время для решения широкого круга комбинаторных оптимизационных научно-технических задач и объединяющих различные либо однотипные алгоритмы, но с различными значениями параметров, в которых преимущества одного алгоритма могут компенсировать недостатки другого. Приводится описание комбинированных биоинспирированных алгоритмов (генетический алгоритм, алгоритмы муравьиных и пчелиных колоний) для реализации криптоанализа шифров перестановок. Определение трудоемкости алгоритмов проводится на основе псевдокода, представляющего реализацию основных операций методов криптоанализа.

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

Научное направление «природные вычисления», объединяющее математические методы, в которых заложен принцип природных механизмов принятия решений, в последние годы получает все более широкое распространение для решения различного круга задач оптимизации, в том числе задач криптоанализа. В течение последних лет были предложены разнообразные схемы эволюционных вычислений: генетический алгоритм, 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 ассиметричных криптосистем с использованием биоинспирированных алгоритмов муравьиных и пчелиных колоний. В [4]–[6] исследована возможность применения методов генетического поиска для реализации криптоанализа блочных криптосистем.

Вопросы разработки и реализации современных стохастических популяционных алгоритмов решения однокритериальной задачи оптимизации рассмотрены в [7], где также предложены пути повышения эффективности этих алгоритмов путем их гибридизации.

Тем не менее следует заметить, что, несмотря на получающее все большее распространение применение биоинспирированных методов и алгоритмов, имитирующих процессы эволюции живой природы, для решения широкого круга оптимизационных задач (в том числе задач криптоанализа) в отечественных и зарубежных публикациях (в том числе в сети Интернет) не приводится каких-либо реальных оценок их трудоемкости, эффективности и сравнительных характеристик, а также строгих математических доказательств корректности реализации (так же как и экспериментальных результатов их применения, поскольку биоинспирированные методы являются вероятностными технологиями, основанными на

имитации процессов живой природы, и их оптимальность может быть доказана только путем проведения экспериментальных исследований).

В связи с этим возникает вопрос о возможности применения комбинированных биоинспирированных алгоритмов для реализации криптоанализа, в частности, о возможности разработки методов, сочетающих основные черты генетических, муравьиных и пчелиных алгоритмов. Одним из основных путей повышения эффективности решения задач глобального поиска является в настоящее время разработка гибридных популяционных алгоритмов (гиперэвристических алгоритмов). Суть гибридизации и интеграции заключается в том, что при выполнении поисковой процедуры производится чередование отдельных процедур, имеющихся в процессе реализации каждого природного алгоритма. Также существенным является то, что данное направление в теории оптимизационных биоинспирированных методов и информационной безопасности является на данный момент практически не исследованным, поэтому разработка комбинированных биоинспирированных методов и их применения для решения задач информационной безопасности и защиты информации, а также оценка их эффективности и возможности применения для решения задач криптоанализа является, несомненно, одной из актуальных проблем.

Отметим, что в настоящее время существует значительное число способов гибридизации оптимизационных алгоритмов, некоторые разновидности классификаций которых приведены в [7].

В данной работе рассматриваются комбинированные биоинспирированные методы, используемые для криптоанализа шифров перестановок и замены, разработанные ранее в [8], [9], и оценивается их эффективность и трудоемкость по сравнению с классическими методами перебора. Ранее задача получения оценок трудоемкости и сравнительных характеристик для классических биоинспирированных методов криптоанализа шифров перестановок и замены рассматривалась в [10], где получены оценки сложности для генетического и муравьиного алгоритмов криптоанализа.

Оценка трудоемкости комбинированного биоинспирированного алгоритма (генетический алгоритм и алгоритм муравьиных колоний). Для получения оценок трудоемкости комбинированных «гиперэвристик», как и ранее в [10], воспользуемся основными правилами получения оценок, описанными в [11], [12]. Отметим, что описание комбинированного биоинспирированного алгоритма криптоанализа на уровне гибридизации вложением приведено в [8], где в качестве популяционного алгоритма используется генетический алгоритм (ГА), а в качестве алгоритма локального поиска – алгоритм муравьиных колоний. Как отмечено в [8], в данном алгоритме операторы 1–4, 9 соответствуют операторам ГА, обеспечивая формирование пространства решений и глобальный поиск, операторы 5–8 соответствуют операторам муравьиного алгоритма и обеспечивают локальный поиск в пространстве решений.

Отметим, что в качестве оптимизационной модели (целевой функции) в [8] предлагается использовать выражение

$$R = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_i C_{ij} X_{ij} \to \max, \qquad (1)$$

где C_{ij} – вероятность того, что за символом в позиции i должен следовать символ в позиции j (вероятность появления в тексте биграммы ij), параметр Q показывает, насколько фрагмент текста носит осмысленный характер, т. е. совпадает со словарным запасом языка. Элементы C_{ij} задаются в виде матрицы размерности $n \cdot n$ (n – число символов текста), $X_{ij} = 1$, если биграмма ij встречается в рассматриваемом тексте и $X_{ij} = 0$ в противном случае.

Для получения оценок трудоемкости, как и ранее в [10], представим данный алгоритм в виде следующего псевдокода (табл. 1), в котором N – размер популяции; n – длина хромосомы популяции, равная длине фрагмента текста; Fer – матрица концентраций феромона; ρ – коэффициент испарения; C – матрица вероятности соседства символов; $N_{\text{расш}}$ – размер расширенной популяции; S – массив для хранения целевых функций маршрутов (популяции индивидуумов); $S_{\max}$, w, v – переменные, используемые для формирования циклов (нахождение максимального элемента, наращивание счетчика результирующих концентраций феромона); $n_{\text{мут}}$ – норма мутации; $F_{\text{Якоб}}$ – трудоемкость функции Якобсена.

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* $N_{расш}$
/*Случайный выбор пары родительских хромосом*/	
8	*I = random(N)*
9	*j = random(N)*
/*Формирование потомка с помощью универсального кроссинговера*/	
10	*For m* =1 *to n*
11	*T = random*(2)
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 путем случайной замены случайного гена в случайной хромосоме */	
15	*For i* = 1 *to* $N_{расш}$
16	*T = random*(100)
17	*If T* <= 100$n_{мут}$ *then j = random(n); Pop(i, j) = random* (алфавит)
18	*Next i*
/*Подсчет критериев оптимальности маршрутов*/	
19	*For k* = 1 *to* $N_{расш}$
20	*S(k)* = 0
21	*For i* = 1 *to n* – 1
22	*S(k) = S(k)+C(Pop(k, i),Pop(k, i + 1))*
23	*Next i*
24	*Next k*
/*Умножение функции качественности на весовой коэффициент*/	
25	*For k* = 1 *to* $N_{расш}$
26	*For i* = 1 *to n*
27	*T=* $F_{Якоб}$*(Pop(k, i))*
28	*S(k) = S(k)T*
29	*Next i*
30	*Next k*
/*Проведение элитного отбора в новую популяцию размера N*/	
31	*For i* =1 *to N*
32	S_{max} = 0
33	*For j* = 1 *to* $N_{расш}$
34	*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*/	
35	*If (S(j) == эталон) then*: *for k* = 1 *to n*; *print Pop(t, k)*; *Next k; Stop*
36	*Next j*
37	*For j* = 1 *to n*
38	*Pop(i, j) = Pop(t, j)*
39	*Next j*
40	*S(t)* = 0
41	*Next i*
/*Формирование матрицы результирующих концентраций феромона*/	
42	*For l* = 1 *to n*
43	*v = алфавит (l)*
44	*w* = 0
45	*For i* = 1 *to n*
46	*For k* = 1 *to N*
47	*If(Pop(k, i) == v) then w = w + S(k)*
48	*Next k*
49	*Fer(l, i) = w*
50	*Next i*
51	*Next l*
/*Проведение испарения феромона*/	
52	*For i* = 1 *to n*
53	*For j* = 1 *to n*
54	*Fer(i, j) = Fer(i, j)*(1 – ρ)
55	*Next j*
56	*Next i*
/*Вычисление вероятностей размещения символов в позиции*/	
57	*For i*=1 *to n*
58	*w* = 0
59	*For k* = 1 *to n*
60	*w = w + Fer(k, i)*
61	*Next k*
62	*For k* = 1 *to n*
63	*Fer(k, i) = Fer(k, i)/w*
64	*Next k*
65	*Next i*
/*Формирование новых *dN* маршрутов*/	
66	*For j = N* + 1 *to N + Nd*
67	*For l* = 1 *to n*
68	*w = random*(1)
69	*v = random(n)*
70	*If(Fer(v, l) <= w) then Pop(j, l) = алфавит(Pop(v, l)); l = n*
71	*Next l*
72	*Next j*
/*Подсчет критериев оптимальности новых *dm* маршрутов*/	
73	*For k = N* + 1 *to N + Nd*
74	*S(k)* = 0
75	*For i* = 1 *to n* – 1
76	*S(k) = S(k) + C(Pop(k, i), Pop(k, i* + 1*))*
77	*Next i*
78	*Next k*
/*Умножение функции качественности на весовой коэффициент для новых *dN* маршрутов*/	
79	*For k = N* + 1 *to N + Nd*
80	*For i* = 1 *to n*
81	*T =* $F_{Якоб}$*(Pop(k, i))*
82	*S(k) = S(k)T*
83	*Next i*
84	*Next k*
/*Выбор *N* маршрутов с лучшими значениями функции качественности*/	
85	*For i* = 1 *to N*
86	S_{max} = 0
87	*For j* = 1 *to N + Nd*
88	*If(S(j) >* S_{max}*) then* S_{max} *= S(j); t = j*
89	*Next j*

Окончание табл. 1

90	*For j* = 1 *to n*
91	*Pop(i, j) = Pop(t, j)*
92	*Next j*
93	*S(t)* = 0
94	*Next i*
/*Проверка достижения заданного числа поколений, если оно достигнуто, вывод популяции маршрутов муравьев и конец работы алгоритма*/	
95	*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*

Представленный алгоритм также является *количественно-зависимым* по трудоемкости, так как его функция трудоемкости зависит от размерности входных данных, а не от их конкретных значений. Как и в [10], данный псевдокод имитирует реализацию комбинированного алгоритма криптоанализа шифров перестановок, описанного в [8], отражает основные операции гибридного алгоритма и позволяет оценить трудоемкость метода на основе основных правил анализа программ как сумму трудоемкостей блоков, следующих друг за другом в алгоритме [12]. Для получения оценок трудоемкости блоков операторов, соответствующих операторам генетического и муравьиного алгоритмов, воспользуемся соотношениями, полученными в [10]. Трудоемкость цикла, выполняемого N раз в количественно-зависимом алгоритме, определим в соответствии с [11], [12] как

$$F_{\text{цикл}} = 1 + 3N + Nf_{\text{тело цикла}},$$

где $f_{\text{тело цикла}}$ – трудоемкость тела цикла.

Применяя методику анализа цикла с операторами 1–5, получим (аналогично [10]): $F_{\text{внутр. цикл } 2\text{–}4} = 1 + 3n + n2$ (в теле цикла содержится 2 операции: присваивание и функция *random*); отсюда $F_{\text{внешн. цикл } 1\text{–}5} = 1 + 3N + NF_{\text{внутр. цикл } 2\text{–}4}$, тогда $F_{\text{цикл } 1\text{–}5} = 1 + 3N + N(1 + 3n + n2) = 1 + 4N + 5Nn = \text{O}(Nn)$.

Трудоемкость оператора 6, очевидным образом составит $F_6 = \text{O}(2)$.

В операторах 7–14 содержатся два вложенных цикла:

$$F_{\text{внутр. цикл } 10\text{–}13} = 1 + 3n + n\left(2 + 0.5 \cdot 1 + 0.5 \cdot 1\right) = 1 + 3n + 3n = 1 + 6n$$

(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, 11 с трудоемкостью 2)

$$F_{\text{внешн. цикл } 7\text{–}14} = 1 + 3(N_{\text{расш}} - N) + (N_{\text{расш}} - N)F_{\text{внутр. цикл } 10\text{–}13} = 1 + 3(N_{\text{расш}} - N) + (N_{\text{расш}} - N)(4 + 1 + 6n) = 1 + 8N_{\text{расш}} - 8N + 6nN_{\text{расш}} - 6nN = \text{O}(nN_{\text{расш}}).$$

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

$$F_{\text{цикл } 15\text{–}18} = 1 + 3N_{\text{расш}} + N_{\text{расш}}(2 + n_{\text{мут}}(2 + 2)) = 1 + 3N_{\text{расш}} + N_{\text{расш}}(4n_{\text{мут}} + 2) = 1 + 5N_{\text{расш}} + 4N_{\text{расш}}n_{\text{мут}} = \text{O}(N_{\text{расш}}n_{\text{мут}}).$$

В цикле операторов 19–24 осуществляется оценка функции пригодности маршрутов, сформированных муравьями, с использованием матрицы вероятности соседства символов C. Трудоемкость внутреннего цикла $F_{\text{цикл } 21\text{–}23} = 1 + 3(n-1) + (n-1)2$, так как в теле цикла (оператор 22) содержится 2 арифметических операции – присваивание и сложение. В этом случае (вследствие наличия оператора 20):

$$F_{\text{цикл } 19\text{–}24} = 1 + 3N_{\text{расш}} + N_{\text{расш}}[1 + 1 + 3(n-1) + (n-1) \cdot 2] = 1 + 5nN_{\text{расш}} = \text{O}(nN_{\text{расш}}).$$

В цикле операторов 25–30 производится умножение функции пригодности маршрутов на весовой коэффициент. При использовании для этой цели функции Якобсена (аналогично [8]–[10]) трудоемкость процесса (вследствие наличия в операторе 27 функции Якобсена и операции присваивания, в операторе 28 операций присваивания и умножения) не должна превысить

$$F_{\text{цикл } 25\text{–}30} = 1 + 3N_{\text{расш}} + N_{\text{расш}}[1 + 3n + n(1 + F_{\text{Якоб}} + 2)] = 1 + 4N_{\text{расш}} + N_{\text{расш}}n(6 + F_{\text{Якоб}}).$$

Как отмечено в [10], 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 $F_{\text{Якоб}} = \text{O}(n)$.

Для подсчета трудоемкости цикла $F_{\text{цикл }31–41}$ определим трудоемкость вложенных циклов. В наихудшем случае, когда массив S идет по возрастанию, трудоемкость оператора 34, выполняемого с вероятностью 1, составит $F_{34} = 2$ (вследствие наличия двух операторов присваивания), трудоемкость цикла в операторе 35 $F_{\text{цикл }35} = 1 + 3n + n \cdot 1 = 1 + 4n$. Вероятность выполнения оператора 35 (вероятность, что функция качества хромосомы равна эталонной) примем равной $F_{\text{Якоб}}$. В этом случае $F_{35} = F_{\text{Якоб}}(1 + 4n + 1)$ (вследствие наличия оператора *stop*). Таким образом, $F_{\text{цикл }33–36} = 1 + 3N_{\text{расш}} + N_{\text{расш}}(2 + F_{\text{Якоб}} \times (1 + 4n + 1))$. Аналогично, $F_{\text{цикл }37–39} = 1 + 3n + n \cdot 1 = 1 + 4n$ (в теле цикла один оператор присваивания). Отсюда трудоемкость всего цикла

$$F_{\text{цикл }31–41} = 1 + 3N + N(1 + 1 + 3N_{\text{расш}} + N_{\text{расш}}(2 + F_{\text{Якоб}}(1 + 4n + 1)) + 1 + 4n + 1) = 1 + 7N + 5NN_{\text{расш}} + 4Nn + 2F_{\text{Якоб}}NN_{\text{расш}} + 4F_{\text{Якоб}}NN_{\text{расш}}n = O(NN_{\text{расш}}n).$$

В цикле, включающем операторы 42–51, осуществляется формирование матрицы результирующих концентраций феромона *Fer* (методика получения данной матрицы описана, например, в [3]). Если принять вероятность того, что символ маршрута муравья равен заданному символу алфавита, равной функции Якобсена, то трудоемкость внутреннего цикла (вследствие наличия оператора присваивания 49)

$$F_{\text{цикл }45–50} = 1 + 3n + n[1 + 3N + NF_{\text{Якоб}}2 + 1] = 1 + 5n + Nn(3 + 2F_{\text{Якоб}}).$$

Трудоемкость внешнего цикла 42–51 в этом случае составит (вследствие наличия двух операторов присваивания 43 и 44)

$$F_{\text{цикл }42–51} = 1 + 3n + n[1 + 1 + 1 + 5n + Nn(3 + 2F_{\text{Якоб}})] = 1 + 6n + n^2(5 + 3N + 2NF_{\text{Якоб}}).$$

Процедура имитации испарения феромона осуществляется в цикле, включающем операторы 52–56 (где ρ – коэффициент испарения). В этом случае (так как оператор 54 включает 3 арифметические операции)

$$F_{\text{цикл }52–56} = 1 + 3n + n(1 + 3n + 3n) = 1 + 4n + 6n^2 = \text{O}(n^2).$$

Процедура формирования матрицы вероятностей размещения символов в позиции осуществляется в цикле, включающем операторы 57–65. Трудоемкость вложенных циклов составит (вследствие того, что операторы 60 и 63 содержат 2 арифметических операции): $F_{\text{цикл }59–61} = 1 + 3n + n \cdot 2 = 1 + 5n$, $F_{\text{цикл }62–64} = 1 + 3n + 2n = 1 + 5n$. Таким образом, трудоемкость вложенных циклов (вследствие наличия оператора присваивания 58)

$$F_{\text{цикл }57–65} = 1 + 3n + n[1 + (1 + 5n) + (1 + 5n)] = 1 + 6n + 10n^2 = \text{O}(n^2).$$

В цикле, включающем операторы 66–72, осуществляется формирование новых dN маршрутов (где d – априорно задаваемый параметр, $d < 1$). Процедура, имитирующая колесо рулетки осуществляется аналогично [10]: параметру w присваивается произвольное значение от 0 до 1 в операторе 68 (вероятность выбора), параметру v присваивается произвольное значение из мощности алфавита n (оператор 69), 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 70). Следовательно, если принять в качестве вероятности выбора оптимального символа значение функции Якобсена (или ее нормированное значение), то трудоемкость внутреннего цикла (операторы 67–71) составит (трудоемкость операторов 68 и 69 равна 2, в операторе 70 содержится 2 оператора присваивания):

$$F_{\text{цикл }67–71} = 1 + 3n + n[2 + 2 + F_{\text{Якоб}} \cdot 2] = 1 + 7n + 2nF_{\text{Якоб}}.$$

В этом случае

$$F_{\text{цикл }66–72} = 1 + 3(N + Nd - N - 1 + 1) + Nd(1 + 7n + 2nF_{\text{Якоб}}) = 1 + 4Nd + nNd(7 + 2F_{\text{Якоб}}).$$

В цикле, содержащем операторы 73–78, осуществляется подсчет критериев оптимальности полученных dN маршрутов с помощью матрицы вероятности соседства символов C. В этом случае трудоемкость внутреннего цикла, включающего операторы 75–77 (так как оператор 76 содержит 2 операции присваивания):

$$F_{\text{цикл }75–77} = 1 + 3(n - 1) + (n - 1)2 = 5n - 4.$$

Трудоемкость внешнего цикла, включающего операторы 73–78 (вследствие наличия оператора 74 с трудоемкостью 1):

$$F_{\text{цикл } 73-78} = 1 + 3\left(N + Nd - N - 1 + 1\right) + \\ + Nd\left(1 + 5n - 4\right) = 1 + 5nNd.$$

Умножение функции качественности на весовой коэффициент для новых dN маршрутов производится в цикле, включающем операторы 79–84. Трудоемкость вложенного цикла, включающего операторы 80–83 (вследствие наличия в операторе 81 операции присваивания и функции Якобсена, а в операторе 82 операций умножения и присваивания) составит:

$$F_{\text{цикл } 80-83} = 1 + 3n + n(1 + F_{\text{Якоб}} + 2) = \\ = 1 + 6n + nF_{\text{Якоб}}.$$

Таким образом, трудоемкость внешнего цикла составит:

$$F_{\text{цикл } 79-84} = 1 + 3(N + Nd - N - 1 + 1) + \\ + Nd(1 + 6n + nF_{\text{Якоб}}) = 1 + 4Nd + nNd(6 + F_{\text{Якоб}}).$$

В цикле, включающем операторы 85–94, осуществляется выбор N маршрутов с лучшими значениями функции качественности. Если массив целевых функций S отсортирован по возрастанию (худший случай), то вероятность выполнения оператора 88 равна 1 и трудоемкость вложенного цикла (операторы 87–89)

$$F_{\text{цикл } 87-89} = 1 + 3(N + Nd) + \\ + (N + Nd)2 = 1 + 5N + 5Nd.$$

Аналогично (так как оператор 91 содержит одну операцию присваивания с трудоемкостью 1):

$$F_{\text{цикл } 90-92} = 1 + 3n + n = 1 + 4n.$$

Таким образом, трудоемкость всего цикла (вследствие наличия оператора 93):

$$F_{\text{цикл } 85-94} = 1 + 3N + N(1 + 1 + 5N + 5Nd + \\ + 1 + 4n + 1) = 1 + 7N + 5N^2(1 + d) + 4Nn.$$

Трудоемкость выполнения циклов, осуществляющих вывод популяции, в операторе 95 определим как

$$F_{\text{цикл } 95} = 1 + 3N + N\left(1 + 3n + n \cdot 1\right) + 1 = \\ = 2 + 4N + 4Nn$$

(вследствие наличия оператора *stop* с трудоемкостью 1). Вероятность выполнения этого оператора определяется количеством итераций *Iterzad*, вероятность блока *then* равна 1/*Iterzad*, вероятность блока *else* с трудоемкостью 1 равна (*Iterzad* – – 1)/*Iterzad*.

В этом случае трудоемкость оператора 95 составит:

$$F_{95} = (1/Iterzad)(2 + 4N + 4Nn) + \\ + \left((Iterzad - 1)/Iterzad\right) \cdot 1 = \\ = \left(1 + 4N + 4Nn + Iterzad\right)/Iterzad.$$

Таким образом, трудоемкость выполнения одной итерации гибридного алгоритма криптоанализа определится как

$$F_{1-95} = 1 + 4N + 5Nn + \ /*операторы\ 1–5*/ \\ + 2 + 1 + 8N_{\text{расш}} - 8N + 6nN_{\text{расш}} - 6nN + \\ /*операторы\ 6–14*/ \\ + 1 + 5N_{\text{расш}} + 4N_{\text{расш}}n_{\text{мут}} + \ /*операторы\ 15–18*/ \\ + 1 + 5nN_{\text{расш}} + /*операторы\ 19–24*/ \\ + 1 + 4N_{\text{расш}} + N_{\text{расш}}n(6 + F_{\text{Якоб}}) + \\ /*операторы\ 25–30*/ \\ + 1 + 7N + 5NN_{\text{расш}} + \ 4Nn + 2F_{\text{Якоб}}NN_{\text{расш}} + \\ + 4F_{\text{Якоб}}NN_{\text{расш}}n + \\ /*операторы\ 31–41*/ \\ + 1 + 6n + n^2(5 + 3N + 2NF_{\text{Якоб}}) + \\ /*операторы\ 42–51*/ \\ + 1 + 4n + 6n^2 + \ /*операторы\ 52–56*/ \\ + 1 + 6n + 10n^2 + /*операторы\ 57–65*/ \\ + 1 + 4Nd + nNd(7 + 2F_{\text{Якоб}}) + \\ /*операторы\ 66–72*/ \\ + 1 + 5nNd + /*операторы\ 73–78*/ \\ + 1 + 4Nd + nNd(6 + F_{\text{Якоб}}) + /*операторы\ 79–84*/ \\ + 1 + 7N + 5N^2(1 + d) + 4Nn + \\ /*операторы\ 85–94*/ \\ + \left(1 + 4N + 4Nn + Iterzad\right)/Iterzad \\ /*оператор\ 95*/.$$

Запишем это выражение в следующем виде:

$$F_{1-95} = 1 + 4N + 5Nn + \ /*операторы\ 1–5*/ \\ + 2 + 1 + 8N_{\text{расш}} - 8N + 6nN_{\text{расш}} - 6nN + \\ /*операторы\ 6–14*/ \\ + 1 + 5N_{\text{расш}} + 4N_{\text{расш}}n_{\text{мут}} + \\ /*операторы\ 15–18*/ \\ + 1 + 5nN_{\text{расш}} + /*операторы\ 19–24*/ \\ + 1 + 4N_{\text{расш}} + 6N_{\text{расш}}n + F_{\text{Якоб}}N_{\text{расш}}n + \\ /*операторы\ 25–30*/ \\ + 1 + 7N + 5NN_{\text{расш}} + \ 4Nn + 2F_{\text{Якоб}}NN_{\text{расш}} + \\ + 4F_{\text{Якоб}}NN_{\text{расш}}n +$$

$$/*\text{операторы } 31\text{–}41*/$$
$$+1+6n+5n^2+3Nn^2+2NF_{\text{Якоб}}n^2+$$
$$/*\text{операторы } 42\text{–}51*/$$
$$+1+4n+6n^2+\ /*\text{операторы } 52\text{–}56*/$$
$$+1+6n+10n^2+/*\text{операторы } 57\text{–}65*/$$
$$+1+4Nd+7nNd+2nNdF_{\text{Якоб}}+$$
$$/*\text{операторы } 66\text{–}72*/$$
$$+1+5nNd+/*\text{операторы } 73\text{–}78*/$$
$$+1+4Nd+6nNd+F_{\text{Якоб}}nNd+$$
$$/*\text{операторы } 79\text{–}84*/$$
$$+1+7N+5N^2+5N^2d+4Nn+/*\text{операторы } 85\text{–}94*/$$
$$+(1+4N+4Nn+Iterzad)/Iterzad$$
$$/*\text{оператор } 95*/.$$

Преобразуя данное выражение, получим, что трудоемкость одной итерации комбинированного алгоритма составит:

$$F_{1\text{–}95}=15+10N+16n+7Nn+17N_{\text{расш}}+$$
$$+17nN_{\text{расш}}+5NN_{\text{расш}}+4N_{\text{расш}}n_{\text{мут}}+$$
$$+21n^2+8Nd+18nNd+3Nn^2+$$
$$+5N^2+5N^2d+3F_{\text{Якоб}}nNd+$$
$$+2NF_{\text{Якоб}}n^2+$$
$$+F_{\text{Якоб}}N_{\text{расш}}n+2F_{\text{Якоб}}NN_{\text{расш}}+$$
$$+4F_{\text{Якоб}}NN_{\text{расш}}n+$$
$$+(1+4N+4Nn+Iterzad)/Iterzad.$$

Окончательно выражение трудоемкости алгоритма примет следующий вид:

$$F_{1\text{–}95}=15+10N+16n+7Nn+17N_{\text{расш}}+$$
$$+17nN_{\text{расш}}+5NN_{\text{расш}}+4N_{\text{расш}}n_{\text{мут}}+$$
$$+21n^2+8Nd+nNd(18+3F_{\text{Якоб}})+$$
$$+Nn^2(3+2F_{\text{Якоб}})+5N^2(1+d)+$$
$$+F_{\text{Якоб}}N_{\text{расш}}n+2F_{\text{Якоб}}N_{\text{расш}}(N+2Nn)+$$
$$+(1+4N+4Nn+Iterzad)/Iterzad. \qquad (2)$$

Таким образом, как следует из выражения (2), трудоемкость итерации гибридного алгоритма криптоанализа (ГА и алгоритм муравьиных колоний) в общем случае определяется квадратичной зависимостью от размера популяции N, размера расширенной популяции $N_{\text{расш}}$ и размера хромосомы n, равной длине фрагмента текста. Поэтому можно считать, что оценка сложности комбинированного алгоритма в общем случае не превосходит оценки сложности муравьиного алгоритма криптоанализа, полученной в [10] (при этом, как и в [10], использовались допущения, что формирование маршрутов в цикле операторов 66–72 всегда возможно (т. е. в матрице *Fer* всегда существует элемент, удовлетворяющий условию $Fer(v, l) <= w$) и что в процессе реализации алгоритма завершение работы производится после просмотра заданного количества итераций *Iterzad*).

Ранее в [8], [9] были приведены соотношения, показывающие, что при использовании комбинированных биоинспирированных алгоритмов вероятность улучшения частичного решения на каждой итерации не может быть меньше вероятности улучшения частичного решения при использовании каждого классического биоинспирированного алгоритма. Таким образом, отмеченные выводы подтверждают целесообразность разработки и использования комбинированных биоинспирированных стратегий и их применения для решения оптимизационных одно- и многоэкстремальных задач.

Отметим также, что применение ГА и алгоритмов муравьиных колоний для криптоанализа шифров замены также отражено в [10]. Как отмечено в [4], [10], в этом случае отличительной особенностью применения биоинспирированных методов криптоанализа является возможность использования самого алгоритма шифрования (или расшифрования) в качестве целевой функции для оценки пригодности ключа, определенного с помощью генетических операций. Криптоанализ шифров замены, 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. Поэтому можно утверждать, что при использовании комбинированных биоинспирированных алгоритмов процесс определения секретного ключа (например, при криптоанализе 2-го типа) зависит не столько от сложности шифрующих преобразований, сколько от самого биоинспирированного алгоритма, который должен обеспечивать достаточное разнообразие генерации ключей. Структурная схема криптоанализа шифров замены с помощью ГА представлена в [10]. Поскольку процесс криптоанализа шифров замены при использовании комбинированных методов производится аналогично, то можно утверждать, что трудоемкость «гибридных» методов криптоанализа также определяется трудоемкостью самого алгоритма шифрования

$F_{\text{шифр}}$. Таким образом, если трудоемкость алгоритма шифрования $F_{\text{шифр}}$ превышает трудоемкость операций самого «гибридного» алгоритма криптоанализа, определяемую выражением (2), то эта оценка $F_{\text{шифр}}$ будет являться верхней оценкой трудоемкости процесса криптоанализа шифров замены с помощью «гибридного» биоинспирированного алгоритма.

Оценка трудоемкости комбинированного биоинспирированного алгоритма (генетический алгоритм и алгоритм пчелиных колоний). Несмотря на широкую область применимости эволюционных методов, они обладают рядом недостатков, отмеченных выше (наличие слепого поиска, приводящего к попаданию в локальный оптимум). В этом плане следует заметить, что одной из последних разработок в области искусственного интеллекта является алгоритм пчел, который используется для нахождения экстремумов сложных многомерных функций [3], [13]–[15].

Как и ранее (аналогично [3], [13]), для решения задачи криптоанализа определяется целевая функция вида, определяемого выражением (1).

Основные операции алгоритма колонии пчел описаны в [3], [9]. Оценка сложности данного алгоритма приведена, например, в [4], [9]. В лучшем случае временная сложность пчелиных алгоритмов T составляет $T \approx \mathrm{O}(n^{\lg n})$, в худшем случае – $T \approx \mathrm{O}(n^3)$.

Пример реализации алгоритма пчелиных колоний для криптоанализа шифров перестановок приведен в [3], [13], для криптоанализа блочных криптосистем на основе определения секретного ключа – в [16]. В связи с этим возникает актуальный вопрос о возможности применения комбинированных биоинспирированных методов для реализации криптоанализа, в том числе о возможности разработки алгоритмов, сочетающих основные черты генетических и пчелиных алгоритмов.

В качестве примера высокоуровневой гибридизации отметим, что комбинированный алгоритм криптоанализа шифров перестановок на уровне гибридизации вложением разработан в [9], где в качестве популяционного алгоритма используется алгоритм пчелиных колоний, а в качестве алгоритма локального поиска – ГА. При этом предполагается, что пространство поиска, в котором размещены символы алфавита шифртекста, представляет собой прямоугольную матрицу A заданного размера $m_1 m_2$.

Как отмечено в [9], в данном алгоритме операторы 1–9, 17–21 соответствуют операторам пчелиного алгоритма, обеспечивая формирование пространства решений и глобальный поиск, операторы 10–16 соответствуют операторам генетического алгоритма и обеспечивают локальный поиск в пространстве решений. В [9] описан и демонстрационный пример реализации данного «гибридного» алгоритма.

Для получения оценок трудоемкости воспользуемся основными правилами оценки, описанными, например, в [11], [12]. Представим данный алгоритм в виде следующего псевдокода (табл. 2), где символ «$\oplus$» означает конкатенацию списков; M – размер популяции пчел; *Iterzad* – количество итераций; n_r – количество агентов-разведчиков; n_f – количество агентов-фуражиров; $\lambda_{\text{макс}}$ – значение максимального размера окрестности; n_b – количество базовых позиций; n_{b1} – количество базовых позиций, формируемых из лучших позиций a^*, найденных на $Iter - 1$ итерации; $E_{\max}$ – максимальная длина списка; n_{rl} – количество агентов-разведчиков, выбирающих случайным образом новые позиции на итерациях 2, 3, …, *Iterzad*; n_{b2} – количество базовых позиций, формируемых из n_{r1} новых лучших позиций, найденных агентами-разведчиками на итерации *Iter*.

Таблица 2

№	Оператор
/*Определение начальных параметров алгоритма */	
1	*M, Iterzad,* n_r, n_f, $\lambda_{\max}$, n_b, n_{b1}, a^*, $E_{\max}$, n_{rl}, n_{b2}
/* Задание номера итерации */	
2	*Iter* = 1
/* Размещение n_r агентов-разведчиков в пространстве поиска */	
3	*For k* =1 *to* n_r
4	$i = random(m_1)$
5	$j = random(m_2)$
6	*A(i, j) = агент-разведчик*
7	*Next k*
/* Определение базовых позиций */	
8	*For k* = 1 *to* n_b
9	*P*1*: i = random*(m_1)
10	$j = random(m_2)$
11	*if(A(i, j) = = агент-разведчик) then A(i, j) = базовая;goto P; else goto P*1
12	*P: Next k*
/* Задание номера агента-фуражира */	
13	*T : f* = 1
/* Выбор базовой позиции и позиции в ее окрестности */	

14	*Z*2: *i = random*(m_1)
15	*j = random*(m_2)
16	*If (A(i, j) = базовая) then goto Z*1 *else goto Z*2
17	*Z*1 *: k = random*(m_1)
18	*m = random*(m_2)
19	*If(A(k, m) = невыбранная) then goto U*1 *else goto Z*1
20	*U*1 *: g = (i–k)+(j–m); If(g <* λ_{max}*) then A(k,m) = = «выбранная»; A(k, m) = A(i, j)* $\oplus$ *A(k, m); else goto Z*1
/* Подсчет целевой функции *R* полученных списков */	
21	*For i* = 1 *to* m_1
22	*For j* = 1 *to* m_2
23	*R(i, j)* = 0
24	*If (A(i, j) = = занятая) then:*
25	*for q* = 1 *to* E_{max}
26	*R(i, j) = R(i, j) +* $C_q(A(i, j)), C_{q+1}(A(i, j))$*; Next q*
27	*Next j*
28	*Next i*
29	*f = f* + 1
30	*If(f <* = n_f*)goto Z*2
/* Проведение операции кроссинговера */	
31	*For q* = 1 *to* n_f *.*$n_{крос}$
32	*V*2 *: k = random*(m_1)
33	*m = random*(m_2)
34	*if(A(k, m) = = содержит_списки) then goto V*1 *else goto V*2
35	*V*1 *: i = random*(m_1)
36	*j = random*(m_2)
37	*if(A(i, j) = = содержит_списки) then goto V3 else goto V*1
/* Формирование потомка с помощью универсального кроссинговера */	
38	*V3 : For g* = 1 *to* $E_{A(i,j)}$
39	*T = random*(2)
40	*If(T = = 0) then* $A_{nomg}(k, m) = A_g(k, m)$*;* $A_{nomg}(i, j) = A_g(i, j)$*; goto V4*
41	*If(T = = 1)then* $A_{nomg}(k, m) = A_g(i, j)$*;* $A_{nomg}(i, j) = A_g(k, l)$*;*
42	*V4 : Next g*
43	*Next q*
/* Проведение операции мутации */	
44	*For i* = 1 *to* m_1
45	*For j* = 1 *to* m_2
46	*If (A(i, j) = = содержит списки) then T = random*(100)*;*
47	*If (T<* = 100$n_{мут}$*)then w = random(*$E_{A(i,j)}$*);*
48	$A_w(i, j)$ = *random*(алфавит)
49	*Next j*
50	*Next i*
/* Подсчет целевых функций вновь полученных индивидуумов */	
51	*For i* = 1 *to* m_1
52	*For j* = 1 *to* m_2

Продолжение табл. 2

53	*R(i, j)* = 0
54	*If(A(i, j) = = потомок) then: for q* = 1 *to* E_{max}
55	*R(i, j) = R(i, j) +* $C_q(A(i, j)), C_{q+1}(A(i, j))$*; Next q*
56	*Next j*
57	*Next i*
/* Умножение на весовой коэффициент целевых функций списков длиной 3 и более символов */	
58	*For i* = 1 *to* m_1
59	*For j* = 1 *to* m_2
60	*If(A(i, j) = = потомок) then:*
61	*If(*$E_{A(i,j)}$*)> = 3) then Q =* $F_{Якоб}(A(i, j))$*; R(i, j) = R(i, j)Q*
62	*Next j*
63	*Next i*
/* Проведение селекции индивидуумов */	
/* Выбор *M* индивидуумов с наибольшей целевой функцией */	
64	*For t* = 1 *to M*
65	R_{max} = 0
66	*For i* = 1 *to* m_1
67	*For j* = 1 *to* m_2
68	*If(R(i, j)>*R_{max}*) then* R_{max} *= R(i, j); i*1 *= i, j*1 *= j*
69	*Next j*
70	*Next i*
71	*A*1*(i*1*, j*1*) = A(i*1*, j*1*)*
72	*R*1*(i*1*, j*1*) = R(i*1*, j*1*)*
73	*A(i*1*, j*1*) = невыбранная*
74	*R(i*1*, j*1*)* = 0
75	*Next t*
/* Занесение массива с лучшими индивидуумами в массив *A(i, j)* */	
76	*For i* = 1 *to* m_1
77	*For j* = 1 *to* m_2
78	*A(i, j) = A*1*(i, j)*
79	*R(i, j) = R*1*(i, j)*
80	*Next j*
81	*Next i*
/* Выбор и сохранение лучшего значения целевой функции и соответствующего списка */	
	R_{max} = 0
82	*For i* = 1 *to* m_1
83	*For j* = 1 *to* m_2
84	*If(R(i, j) >* R_{max}*) then* R_{max} *= R(i, j); i*1 *= i; j*1 *= j;* $R^*(Iter) = R_{max}$
85	*Next j*
86	*Next i*
/* Вывод списка с лучшим значением целевой функции и значения целевой функции на итерации *Iter* */	
87	*Print* $R^*(Iter)$
88	*For q* = 1 *to* $E_{A(i,j)}$
89	*Print* $A_q(i1, j1)$
90	*Next q*
/* Сохранение оптимального значения целевой функции */	
91	*If(Iter = = 1)then goto C*1
92	*If(*$R^*(Iter) > R^*(Iter – 1)$*) then* $R^*_{opt} = R^*(Iter)$ *else* $R^*_{opt} = R^*(Iter – 1)$

Окончание табл. 2

93	*S*1 *: If(Iter < Iterzad)then goto C*1
/* Вывод лучшего значения целевой функции */	
94	*Else print* $R^*_{\text{опт}}$ *; stop*
/* Наращивание счетчика итераций */	
95	*C*1 *: Iter = Iter* + 1
/* Формирование множества A_{b1} базовых позиций для следующей итерации */	
96	*For k* = 1 *to* n_{b1}
97	*I*2*:i = random*(m_1)
98	*j = random*(m_2)
99	*If(A(i, j) = = выбранная_на_предыдущей_итерации) then A(i, j) = = базовая; goto I*1*; else goto I*2
100	*I*1*: Next k*
/* Размещение n_{rl} агентов-разведчиков случайным образом в пространстве поиска */	
101	*For k* = 1 *to* n_{rl}
102	*i = random*(m_1)
103	*j = random*(m_2)
104	*A(i, j) = агент-разведчик*
105	*Next k*
/* Включение во множество A_{b2} n_{b2} позиций из множества n_{rl} позиций */	
106	*For k* = 1 *to* n_{b2}
107	*Y*2*: i = random*(m_1)
108	*j = random*(m_2)
109	*If(A(i, j) = = агент-разведчик then A(i, j) = базовая; goto Y*1*; else goto Y*2
110	*Y*1*: Next k*
111	*goto T*

Таким образом, данный псевдокод, как и ранее, имитирует реализацию комбинированного алгоритма криптоанализа шифров перестановок, отражает сущность основных операций гибридного алгоритма и позволяет оценить трудоемкость алгоритма на основе основных правил анализа программ как сумму трудоемкостей блоков, следующих друг за другом в алгоритме [11], [12]. Можно считать, что данный алгоритм также является *количественно-зависимым* по трудоемкости, так как его функция трудоемкости зависит только от размерности, а не от конкретных значений входных данных. Трудоемкость блока 1 (ввод входных данных) примем равной $F_1 = \text{O}(10)$, трудоемкость блока 2 (задание номера итерации с помощью операции присваивания) $F_2 = \text{O}(1)$. В цикле операторов 3–7 осуществляется операция размещения n_r агентов-разведчиков в пространстве поиска. Для определения трудоемкости циклов, выполняемых N раз, воспользуемся приведенным ранее соотношением $F_{\text{цикл}} = 1 + 3N + Nf_{\text{тело цикла}}$ [11], [12]. В этом случае $F_{3-7} = 1 + 3n_r + n_r 5$, так как трудоемкость операторов 4 и 5 равна O(2), в операторе 6 осуществляется присвоение символу пространства поиска метки «агент-разведчик» (для этого может быть использовано понятие «нитка дискрета», аналогичное введенному в [17]).

В цикле операторов 8–12 осуществляется выбор n_b базовых позиций из n_r позиций, определенных агентами-разведчиками. В этом случае, принимая вероятность выполнения оператора 11 $\frac{n_r}{m_1 m_2}$, трудоемкость выполнения оператора 11 составит (в блоке *then* два оператора с трудоемкостью 1, в блоке *else* один оператор безусловного перехода): $F_{11} = \frac{n_r}{m_1 m_2} 2 + \left(1 - \frac{n_r}{m_1 m_2}\right) 1$.

Трудоемкость всего цикла составит $F_{8-12} = 1 + 3n_b + n_b \left[2 + 2 + \frac{n_r}{m_1 m_2} 2 + \left(1 - \frac{n_r}{m_1 m_2}\right) 1\right]$, т. е. $F_{8-12} = 1 + 3n_b + n_b \left(5 + \frac{n_r}{m_1 m_2}\right)$.

Трудоемкость оператора 13, очевидным образом, составит $F_{13} = \text{O}(1)$.

Операторы 14–20 используются для последовательного выбора агентов-фуражиров и позиций в их окрестности.

Трудоемкость операторов 14 и 15 (произвольный выбор позиции в пространстве поиска) равна $F_{14} = \text{O}(2)$, $F_{15} = \text{O}(2)$.

В операторе 16 осуществляется проверка, является ли выбранная позиция базовой. Его трудоемкость (полагая вероятность нахождения базовой позиции $n_b / (m_1 m_2)$) $F_{16} = n_b / (m_1 m_2) 1 + \left(1 - \frac{n_b}{m_1 m_2}\right) 1$, т. е. $F_{16} = 1$.

Если выбранная позиция является базовой, то производится произвольный выбор позиции в пространстве поиска (операторы 17, 18), при этом $F_{17} = \text{O}(2)$, $F_{18} = \text{O}(2)$.

Если данная позиция не выбиралась ранее на данной итерации, то производится переход на определение расстояния от нее до базовой позиции, в противном случае – возврат на произвольный выбор позиции (оператор 19, полагая веро-

ятность его выполнения $1-n_b$, получим, что его трудоемкость равна $F_{19}=1$).

Если расстояние до базовой позиции меньше значения максимального размера окрестности $\lambda_{\max}$, то данная позиция считается выбранной (вероятность этого примем равной $\frac{\lambda_{\max}\lambda_{\max}}{m_1 m_2}$, т. е. отношение площади допустимой окрестности к площади пространства поиска) и определяется соответствующее частичное решение-список, в противном случае производится возврат на произвольный выбор позиции. Тогда $F_{20}=4+\frac{\lambda_{\max}\lambda_{\max}}{m_1 m_2}3+\left(1-\frac{\lambda_{\max}\lambda_{\max}}{m_1 m_2}\right)1$, т. е. $F_{20}=5+\frac{2\lambda_{\max}\lambda_{\max}}{m_1 m_2}$.

В операторах 21–27 производится подсчет целевой функции полученных списков.

Трудоемкость оператора 26 примем равной $F_{26}=\mathrm{O}(2)$, так как он содержит 2 операции – присваивание и сложение. Тогда трудоемкость внутреннего цикла (операторы 25–26, $E_{\max}$ – максимальная длина списка) составит $F_{25\text{–}26}=1+3E_{\max}+E_{\max}\cdot 2$. Принимая вероятность выполнения операторов 25–26 равной $\frac{2n_f}{m_1 m_2}$ (отношение удвоенного числа агентов-фуражиров к площади пространства поиска), получим, что $F_{25\text{–}26}=\frac{2n_f}{m_1 m_2}(1+3E_{\max}+E_{\max}2)$. Трудоемкость внутреннего цикла (операторы 22–27) в этом случае составит $F_{22\text{–}27}=1+3m_2+$ $+m_2\left[1+\frac{2n_f}{m_1 m_2}(1+3E_{\max}+E_{\max}2)\right]$ (вследствие наличия оператора 23 с трудоемкостью $F_{23}=1$). Тогда общая трудоемкость операторов 21–28 составит

$$F_{21\text{–}28}=1+3m_1+m_1\times$$
$$\times\left\{1+3m_2+m_2\left[1+\frac{2n_f}{m_1 m_2}(1+3E_{\max}+E_{\max}2)\right]\right\}.$$

В операторе 29 осуществляется наращивание номера агента-фуражира, $F_{29}=\mathrm{O}(2)$. Если не все агенты-фуражиры осуществили нахождение позиций в пространстве поиска, производится возврат на нахождение базовой позиции и позиции в ее окрестности (оператор 30), максимальное число выполнения данного оператора n_f. Поэтому вероятность его выполнения примем равной 1, и он содержит один оператор перехода. Таким образом, $F_{30}=1$.

Проведение операции кроссинговера осуществляется с помощью операторов 31–43. В операторах 32 и 33 производится выбор произвольной занятой позиции, их трудоемкость равна $F_{32}=\mathrm{O}(2)$, $F_{33}=\mathrm{O}(2)$. Так как в операторе 34 блоки *then* и *else* содержат по одному оператору безусловного перехода, то $F_{34}=\mathrm{O}(1)$. Аналогично $F_{35}=\mathrm{O}(2)$, $F_{36}=\mathrm{O}(2)$, $F_{37}=\mathrm{O}(1)$ (выбор второго родителя для кроссинговера, при этом предполагается, что кроссинговер между родителями возможен).

С помощью операторов цикла 38–42 осуществляется формирование потомков с помощью операции универсального кроссинговера при длине списков $E_{A(i,j)}$. Выбор значения переменной для формирования маски производится в операторе 39, $F_{39}=\mathrm{O}(2)$. Формирование гена потомка в зависимости от гена маски производится в операторе 40. Его трудоемкость можно определить как $F_{40}=\frac{1}{2}(1+1+1)=\frac{3}{2}$. Аналогично $F_{41}=\frac{1}{2}(1+1)=1$. В этом случае трудоемкость операторов цикла 38–42 определится как $F_{38\text{–}42}=1+3E_{A(i,j)}+E_{A(i,j)}(2+1.5+1)=1+7.5\times$ $\times E_{A(i,j)}$.

Таким образом, трудоемкость операции формирования потомков при норме кроссинговера $n_{\text{крос}}$ ($n_f n_{\text{крос}}$ потомков) составит (цикл операторов 31–43)

$$F_{31\text{–}43}=1+3n_f n_{\text{крос}}+n_f n_{\text{крос}}\times$$
$$\times(2+2+1+2+2+1+1+7.5E_{A(i,j)}).$$

С помощью операторов цикла 44–50 осуществляется проведение оператора точечной мутации. Принимая вероятность того, что позиция пространства поиска содержит список, как и ранее, $\frac{2n_f}{m_1 m_2}$, получим, что $F_{46}=\frac{2n_f}{m_1 m_2}2=\frac{4n_f}{m_1 m_2}$, так как оператор 46 содержит два элементарных операции (присваивание и функция *random*).

Принимая вероятность выполнения оператора 47 равной $n_{\text{мут}}$, получим, что $F_{47} = n_{\text{мут}} 2$ (оператор содержит операции присваивания и функция *random*). Аналогично $F_{48} = 2$ (выбор произвольного символа алфавита для позиции $A_W(i, j)$). В этом случае трудоемкость внутреннего цикла (операторы 45–49) составит $F_{45-49} = 1 + 3m_2 + m_2\left(\frac{4n_f}{m_1 m_2} + 2n_{\text{мут}} + 2\right)$. Трудоемкость внешнего цикла (операторы 44–50) $F_{44-50} = 1 + 3m_1 + m_1\left[1 + 3m_2 + m_2\left(\frac{4n_f}{m_1 m_2} + 2n_{\text{мут}} + 2\right)\right]$.

С помощью операторов цикла 51–57 производится подсчет целевых функций потомков. Принимая вероятность появления потомка равной $\frac{n_f n_{\text{крос}}}{M}$ (отношение числа потомков к общему числу популяции пчел), получим, что трудоемкость внутреннего цикла $F_{54-55} = \frac{n_f n_{\text{крос}}}{M} \times (1 + 3E_{\max} + E_{\max} 2)$. В этом случае (вследствие наличия оператора 53) $F_{52-56} = 1 + 3m_2 + m_2\left[1 + \frac{n_f n_{\text{крос}}}{M}(1 + 3E_{\max} + E_{\max} 2)\right]$. Трудоемкость операторов цикла 51–57: $F_{51-57} = 1 + 3m_1 + m_1\left\{1 + 3m_2 + m_2\left[1 + \frac{n_f n_{\text{крос}}}{M}(1 + 3E_{\max} + E_{\max} 2)\right]\right\}$.

В операторах цикла 58–63 производится умножение целевой функции полученных частичных решений на весовой коэффициент. Учитывая, что списки длиной 3 появляются на итерациях 3, …, *Iterzad*, вероятность того, что длина списка $E_{A(i,j)} >= 3$ примем равной $P(E_{A(i,j)} >= 3) = \frac{Iterzad - 2}{Iterzad}$. В этом случае трудоемкость операторов 60–61 составит $F_{60-61} = \frac{n_f n_{\text{крос}}}{M} \times \left[\frac{Iterzad - 2}{Iterzad}(3 + F_{\text{Якоб}})\right]$. Трудоемкость операторов цикла 59–62: $F_{59-62} = 1 + 3m_2 + m_2 \times \left\{\frac{n_f n_{\text{крос}}}{M}\left[\frac{Iterzad - 2}{Iterzad}(3 + F_{\text{Якоб}})\right]\right\}$. В этом случае

$$F_{58-63} = 1 + 3m_1 + m_1 \times \left\langle 1 + 3m_2 + m_2\left\{\frac{n_f n_{\text{крос}}}{M}\left[\frac{Iterzad - 2}{Iterzad}(3 + F_{\text{Якоб}})\right]\right\}\right\rangle.$$

С помощью операторов цикла 64–75 осуществляется выбор M элементов популяции с наибольшим значением целевой функции из популяции родителей и потомков. Для этого используются вспомогательные массивы $A1(i, j)$ и $R1(i, j)$ для сохранения оптимального значения целевой функции и списков с лучшим значением целевой функции. Если в массиве значений целевой функции $R(i, j)$ элементы расположены по возрастанию, то трудоемкость оператора 68 составит (вероятность, что $R(i, j) > R_{\text{макс}}$ равна 1, в блоке *then* 3 операции присваивания) $F_{68} = 1 \cdot 3 = 3$. Тогда трудоемкость операторов цикла 67–69: $F_{67-69} = 1 + 3m_2 + m_2 3 = 1 + 6m_2$, а трудоемкость операторов цикла 6–70: $F_{66-70} = 1 + 3m_1 + m_1(1 + 6m_2) = 1 + 4m_1 + 6m_1 m_2$.

Операторы 71 и 72 используются для сохранения значений элементов массивов R и A в вспомогательных массивах (максимального значения целевой функции и соответствующего списка) и освобождения позиции с оптимальным списком и значением его целевой функции для последующего поиска нового оптимального элемента-списка. Их трудоемкость $F_{71} = 1$, $F_{72} = 1$, $F_{73} = 1$, $F_{74} = 1$. В этом случае трудоемкость операторов цикла 64–75 составит (вследствие наличия оператора присваивания 65) $F_{64-75} = 1 + 3M + M(1 + 1 + 4m_1 + 6m_1 m_2 + 4)$, т. е. $F_{64-75} = 1 + 9M + M(4m_1 + 6m_1 m_2)$.

С помощью операторов цикла 76–81 производится обновление массива списков $A(i, j)$ (в нем сохраняются только M лучших списков из массива $A1(i, j)$) и $R(i, j)$ (в нем сохраняются только M лучших значений целевых функций списков из массива $R1(i, j)$). Трудоемкость операторов цикла 77–80 составит $F_{77-80} = 1 + 3m_2 + m_2 \cdot 2$, а трудоемкость операторов цикла 76–81 составит $F_{76-81} = 1 + 3m_1 + m_1(1 + 5m_2)$.

С помощью операторов цикла 82–86 производится поиск максимального элемента $R_{\max}$ в массиве целевых функций $R(i, j)$ и запоминание его координат с помощью переменных (i_1, j_1). Если в массиве значений целевой функции $R(i, j)$ элемен-

ты расположены по возрастанию, то трудоемкость оператора 84 составит (вероятность, что $R(i, j) > R_{\max}$ равна 1, в блоке *then* 4 операции присваивания) $F_{84} = 1 \cdot 4 = 4$. Трудоемкость операторов цикла 83–85 составит $F_{83–85} = 1 + 3m_2 + m_2 4 = 1 + 7m_2$, а операторов цикла 82–86: $F_{82–86} = 1 + 3m_1 + m_1(1 + 7m_2) = 1 + 4m_1 + 7m_1m_2$.

С помощью операторов 87–90 осуществляется вывод найденного лучшего значения целевой функции на данной итерации *Iter* и соответствующего списка. В этом случае $F_{87} = 1$, если вывод списка производится поэлементно в цикле, то трудоемкость цикла составит ($E_{A(i,j)}$ – длина списка) $F_{88–90} = 1 + 3E_{A(i, j)} + E_{A(i, j)}1 = 1 + 4E_{A(i, j)}$.

С помощью операторов 91–95 производится сохранение найденного на данной итерации оптимального значения целевой функции R^* и наращивание счетчика итераций. Вероятность выполнения оператора 91 (что данная итерация первая) равна $\frac{1}{Iterzad}$, поэтому $F_{91} = \frac{1}{Iterzad}$. В случае, если оптимальные значения R^* на каждой итерации увеличиваются, вероятность выполнения оператора 92 равна 1 и его трудоемкость $F_{92} = 1$. Аналогично вероятность выполнения операторов 93 и 94 (проверка, достигнуто ли заданное количество итераций) равна $\frac{Iterzad - 1}{Iterzad}$, поэтому $F_{93–94} = \frac{Iterzad - 1}{Iterzad}1 + \frac{1}{Iterzad}2 = \frac{Iterzad + 1}{Iterzad}$. Трудоемкость оператора 95 $F_{95} = 2$.

В цикле операторов 96–100 производится выбор n_{b1} лучших позиций, найденных на $Iter - 1$ итерации. Трудоемкость операторов 97 и 98 $F_{97} = 2$, $F_{98} = 2$. Принимая вероятность того, что найдена лучшая позиция из $Iter - 1$ итерации $\frac{M}{m_1m_2}$, получим, что

$$F_{97–99} = 2 + 2 + \frac{M}{m_1m_2}2 + \frac{m_1m_2 - M}{m_1m_2}1 = 4 + \frac{m_1m_2 + M}{m_1m_2}.$$

В этом случае трудоемкость всего цикла $F_{96–100} = 1 + 3n_{b1} + n_{b1}\left(4 + \frac{m_1m_2 + M}{m_1m_2}\right)$.

В цикле операторов 101–105 производится размещение n_{rl} агентов-разведчиков в пространстве поиска. Таким образом, $F_{101–105} = 1 + 3n_{rl} + n_{rl} \cdot 5$.

В цикле операторов 106–110 производится выбор n_{b2} базовых позиций, формируемых из n_{rl} новых лучших позиций, найденных агентами-разведчиками на итерации *Iter*. Трудоемкость операторов 107 и 108 $F_{107} = 2$, $F_{108} = 2$, оператора 109: $F_{109} = \frac{n_{rl}}{m_1m_2}2 + \frac{m_1m_2 - n_{rl}}{m_1m_2}1$. В этом случае трудоемкость всего цикла составит

$$F_{106–110} = 1 + 3n_{b2} + n_{b2} \times \left(2 + 2 + \frac{n_{rl}}{m_1m_2}2 + \frac{m_1m_2 - n_{rl}}{m_1m_2}1\right) = 1 + 3n_{b2} + n_{b2}\left(4 + \frac{m_1m_2 - n_{rl}}{m_1m_2}\right).$$

Трудоемкость оператора $F_{111} = 1$. Таким образом, трудоемкость выполнения одной итерации гибридного алгоритма криптоанализа (генетический алгоритм и алгоритм пчелиных колоний) определится как

$$F_{1–111} = 10 + 1 + 1 + 3n_{r\cdot} + n_r \cdot 5 + 1 + 3n_b + n_b\left(5 + \frac{n_r}{m_1m_2}\right) +$$

/*операторы 1–2, 3–7,8–12*/

$$+ 1 + 2 + 2 + 1 + 2 + 2 + 1 + 5 + \frac{2\lambda_{\max}\lambda_{\max}}{m_1m_2} +$$

/*операторы 13–20*/

$$+ 1 + 3m_1 + m_1\left\{1 + 3m_2 + m_2 \times \left[1 + \frac{2n_f}{m_1m_2}(1 + 3E_{\max} + E_{\max}2)\right]\right\} +$$

/*операторы 21–28*/

$$+ 2 + 1 +$$ /*операторы 29–30*/

$$+ 1 + 3n_f n_{\text{крос}} + n_f n_{\text{крос}}(11 + 7.5E_{A(i,j)}) +$$

/*операторы 31–43*/

$$+ 1 + 3m_1 + m_1\left[1 + 3m_2 + m_2\left(\frac{4n_f}{m_1m_2} + 2n_{\text{мут}} + 2\right)\right] +$$

$$+ 1 + 3m_1 + m_1\left[1 + 3m_2 + m_2\left(\frac{4n_f}{m_1m_2} + 2n_{\text{мут}} + 2\right)\right] +$$

/*операторы 44–50*/

$$+ 1 + 3m_1 + m_1\left\{1 + 3m_2 + m_2\left[1 + \frac{n_f n_{\text{крос}}}{M}(1 +\right.\right.$$

$$+3E_{\max}+E_{\max}2]\}$$
$$/*\text{операторы } 51–57*/$$
$$+1+3m_1+m_1\langle 1+3m_2+m_2\times$$
$$\times\left\{\frac{n_f n_{\text{крос}}}{M}\left[\frac{Iterzad-2}{Iterzad}(3+F_{\text{Якоб}})\right]\right\}\rangle+$$
$$/*\text{операторы } 58–63*/$$
$$+1+9M+M(4m_1+6m_1m_2)+$$
$$/*\text{операторы } 64–75*/$$
$$+1+3m_1+m_1(1+5m_2)+1+4m_1+7m_1m_2+$$
$$/*\text{операторы } 76–81,\ 82–86*/$$
$$+1+1+4E_{A(i,j)}+\frac{1}{Iterzad}+1+\frac{Iterzad+1}{Iterzad}+2+$$
$$/*\text{операторы } 87–95*/$$
$$+1+3n_{b1}+n_{b1}\left(4+\frac{m_1m_2+M}{m_1m_2}\right)+1+3n_{rl}+n_{rl}5+$$
$$/*\text{операторы } 96–105*/$$
$$+1+3n_{b2}+n_{b2}\left(4+\frac{m_1m_2+n_{rl}}{m_1m_2}\right)+1$$
$$/*\text{операторы } 106–111*/.$$

Запишем это выражение в следующем виде:

$$F_{1–111}=13+8n_r+8n_b+\frac{n_bn_r}{m_1m_2}+$$
$$/*\text{операторы } 1–12*/$$
$$+16+\frac{2\lambda_{\max}\lambda_{\max}}{m_1m_2}+\ /*\text{операторы } 13–20*/$$
$$+4+4m_1+4m_1m_2+2n_f+10n_fE_{\max}+$$
$$/*\text{операторы } 21–30*/$$
$$+1+14n_fn_{\text{крос}}+7.5n_fn_{\text{крос}}E_{A(i,j)}+$$
$$/*\text{операторы } 31–43*/$$
$$+1+4m_1+5m_1m_2+4n_f+2n_{\text{мут}}m_1m_2+$$
$$/*\text{операторы } 44–50*/$$
$$+1+4m_1m_2+4m_1m_2+$$
$$+\frac{m_1m_2n_fn_{\text{крос}}+5m_1m_2E_{\max}n_fn_{\text{крос}}}{M}+$$
$$/*\text{операторы } 51–57*/$$
$$/*\text{операторы } 58–63*/$$
$$+1+4m_1+3m_1m_2+$$
$$+\frac{m_1m_2n_fn_{\text{крос}}(3Iterzad-6+F_{\text{Якоб}}(Iterzad-2))}{M\ Iterzad}+$$
$$+1+9M+4m_1M+6m_1m_2M+$$
$$/*\text{операторы } 64–75*/$$
$$+2+8m_1+12m_1m_2+/*\text{операторы } 76–86*/$$
$$+6+4E_{A(i,\,j)}+\frac{2}{Iterzad}+/*\text{операторы } 87–95*/$$
$$+2+8n_{b1}+\frac{n_{b1}M}{m_1m_2}+8n_{rl}+/*\text{операторы } 96–105*/$$
$$+2+8n_{b2}+\frac{n_{b2}n_{rl}}{m_1m_2}\ /*\text{операторы } 106–111*/.$$

Преобразуя данное выражение, получим, что трудоемкость одной итерации комбинированного алгоритма составит:

$$F_{1–111}=13+8n_r+8n_b+\frac{n_bn_r}{m_1m_2}+$$
$$37+\frac{2\lambda_{\max}\lambda_{\max}}{m_1m_2}+24m_1+28m_1m_2+6n_f+$$
$$+4E_{A(i,j)}+\frac{2}{Iterzad}+$$
$$+8(n_{b1}+n_{rl}+n_{b2})+\frac{n_{b1}M+n_{b2}n_{rl}}{m_1m_2}+$$
$$+10n_fE_{\text{макс}}+n_fn_{\text{крос}}(14+7.5E_{A(i,j)})+$$
$$+2n_{\text{мут}}m_1m_2+M(9+m_1(4+6m_2))+$$
$$+\{m_1m_2n_fn_{\text{крос}}[Iterzad(4+5E_{\text{макс}}+$$
$$+F_{\text{Якоб}})-2(3+F_{\text{Якоб}})]\}/M\ Iterzad.$$

Так как значение $E_{\text{макс}}$ является верхней оценкой длины списков $E_{A(i,\ j)}$, находящихся в позициях пространства поиска $A(i, j)$, то отсюда можно сделать вывод, что трудоемкость всего комбинированного алгоритма криптоанализа определится выражением

$$F_{\text{крипт}}=13+8n_r+8n_b+\frac{n_bn_r}{m_1m_2}+$$
$$+Iterzad\left(37+\frac{2\lambda_{\max}\lambda_{\max}}{m_1m_2}+24m_1+\right.$$
$$+28m_1m_2+6n_f+4E_{\max}+$$
$$+8(n_{b1}+n_{rl}+n_{b2})+\frac{n_{b1}M+n_{b2}n_{rl}}{m_1m_2}+10n_fE_{\max}+$$
$$+n_fn_{\text{крос}}(14+7.5E_{\max})+$$
$$+2n_{\text{мут}}m_1m_2+M(9+m_1(4+6m_2)))+2+$$
$$+\{m_1m_2n_fn_{\text{крос}}[Iterzad(4+5E_{\max}+$$
$$\left.+F_{\text{Якоб}})-2(3+F_{\text{Якоб}})]\}/M. \qquad (3)$$

Таким образом, как показывает выражение (3), трудоемкость комбинированного алгоритма криптоанализа (генетический алгоритм и алгоритм пчелиных колоний) определяется количеством агентов-разведчиков n_r, количеством базовых позиций n_b, количеством агентов-фуражиров

n_f, а также обратно пропорциональна размеру пространства поиска $m_1 m_2$ и размеру популяции пчел M при фиксированном размере пространства поиска и параметрах n_f, $n_{\text{крос}}$, *Iterzad*, в то время как оценки трудоемкости биоинспирированных алгоритмов криптоанализа, полученные в [10], а также гибридного алгоритма (генетический алгоритм и алгоритм муравьиных колоний) обладают прямо пропорциональной зависимостью от параметров алгоритма, что в общем случае может быть связано с фиксированным размером пространства поиска для данного гибридного алгоритма (генетический алгоритм и алгоритм пчелиных колоний).

Полученная оценка трудоемкости позволяет в общем случае также приближенно оценить сложность пчелиного алгоритма криптоанализа (является его верхней оценкой), описанного в [3] (в этом случае могут отсутствовать оценки сложности генетических операций). Так как оценки сложности генетических операций (кроссинговер, мутация) имеют линейную зависимость, то можно утверждать, что оценки сложности пчелиного алгоритма обладают зависимостью от значений входных параметров, аналогичной зависимости, имеющей место в выражении (3).

Таким образом, в данной статье была рассмотрена актуальная задача – определение эффективности и сравнительных характеристик комбинированных биоинспирированных алгоритмов для решения комбинаторных оптимизационных задач криптоанализа. На основе приведенного псевдокода, представляющего реализацию основных операций алгоритмов криптоанализа, получены расчетные формулы, определяющие трудоемкость итерации алгоритма. Показано, что оценка сложности комбинированного алгоритма (ГА и алгоритм муравьиных колоний) в общем случае не превосходит оценки сложности муравьиного алгоритма криптоанализа, полученной в [10], а также, что сложность гибридного алгоритма (ГА и алгоритм пчелиных колоний) обратно пропорциональна размеру пространства поиска $m_1 m_2$ и размеру популяции пчел M при фиксированном размере пространства поиска и параметрах n_f, $n_{\text{крос}}$, *Iterzad*. В то же время оценки трудоемкости биоинспирированных алгоритмов криптоанализа, полученные в [10], а также гибридного алгоритма (генетический алгоритм и алгоритм муравьиных колоний) обладают прямо пропорциональной зависимостью от параметров алгоритма.

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 / КВВУ. Краснодар, 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. Донского гос. техн. ун-та. 2015. № 3(82). С. 65–72.

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

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

8. 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.

9. Чернышев Ю. О., Сергеев А. С. Применение комбинированных биоинспирированных стратегий (генетический алгоритм и алгоритм пчелиных колоний) для реализации криптоанализа классических шифров перестановок // Инж. вестн. Дона. 2017. № 4. URL: http://ivdon.ru/ru/magazine/archive/n4y2017/4518 (дата обращения: 19.06.2019).

10. Чернышев Ю. О., Сергеев А. С., Рязанов А. Н. Оценка эффективности и сравнительные характеристики бионспирированных алгоритмов криптоанализа // Изв. СПбГЭТУ «ЛЭТИ». 2018. № 9. С. 33–44.

11. Трудоемкость алгоритмов и временные оценки. URL: http://th-algoritmov.narod.ru/5.htm (дата обращения: 19.06.2019).

12. Анализ алгоритмов. Сравнительные оценки алгоритмов. URL: http://fsc.bsu.by/wp-content/uploads/2015/12/AiSD-Tema-Analiz-algoritmov.pdf (дата обращения: 19.06.2019).

13. Исследование возможности применения бионических методов пчелиных колоний для реализации криптоанализа классических шифров перестановок / Ю. О. Чернышев, А. С. Сергеев, Е. О. Дубров, А. Н. Рязанов // Вестн. ДГТУ. 2014. Т. 14, № 1(76). С. 62–75.

14. Алгоритм пчел для оптимизации функции. URL: http://jenyay.net/Programming/Bees (дата обращения: 19.06.2019).

15. Алгоритм пчел для оптимизации функции. URL: http://lit999.narod.ru/soft/ga/index.html (дата обращения: 19.06.2019).

16. Сергеев А. С., Рязанов А. Н., Дубров Е. О. Применение алгоритмов пчелиных колоний для реализации криптоанализа блочных методов шифрования // Инж. вестн. Дона. 2016. № 2. URL: http://ivdon.ru/ru/magazine/archive/n2y2016/3621 (дата обращения: 19.06.2019).

17. Сергеев А. С. Исследование и разработка методов трассировки проводящих покрытий БИС на основе стратегии эволюционного поиска: дис. ... канд. техн. наук / ИЦ ДГТУ. Ростов н/Д., 2000. 171 с.

Yu. O. Chernyshev, A. S. Sergeyev
Don State technical university

P. A. Panasenko
Krasnodar higher military school of a name of the General S. M. Shtemenko

ASSESSMENT OF EFFICIENCY AND COMPARATIVE CHARACTERISTICS OF THE COMBINED BIOINSPIRED CRYPTANALYSIS ALGORITHMS

Is devoted to the solution of a relevant task of cryptanalysis with use of new model of optimizing strategy – the combined bioinspired algorithms. The article deals with the problem of determining the effectiveness of combined bioinspired algorithms used in recent years to solve a wide range of combinatorial optimization scientific and technical problems and combining different or similar algorithms, but with different values of the parameters in which the advantages of one algorithm can compensate for the shortcomings of another. Application of the combined bioinspired algorithms (a genetic algorithm, algorithms of ant and bee colonies) for realization of cryptanalysis of permutation ciphers is considered. Determination of the complexity of the algorithms is carried out on the basis of the pseudo-code, which represents the implementation of the basic operations of cryptanalysis methods.

Cryptanalysis, bioinspired algorithms, genetic algorithm, ant colony algorithm, bee colony algorithm, algorithm complexity, efficiency, permutation cipher