ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 681.5.015.23

АЛГОРИТМ ЛЕТУЧИХ МЫШЕЙ (BA) ДЛЯ ЗАДАЧИ ГЛОБАЛЬНОЙ БЕЗУСЛОВНОЙ ОПТИМИЗАЦИИ

Лагунова А.Д.
ФГБУН Институт автоматики и процессов управления
Дальневосточного отделения Российской академии наук
Владивосток, Россия
690041, Владивосток, ул. Радио, 5
e-mail: schimka_06@mail.ru

Аннотация

В данной работе проведено исследование алгоритма летучих мышей (Bat Algorithm, BA) на примере задачи глобальной безусловной оптимизации. Было изучено влияние размера популяций и количества итераций на эффективность алгоритма, а также выявлены наиболее оптимальные их значения для 2- и 3-мерных функций. Для исследования эффективности и сравнительного анализа алгоритма был разработан программный модуль, в котором также реализованы такие методы, как: полный перебор, метод сеток, GWO, параллельный полный перебор, параллельный метод сеток. В качестве критериев оценки были выбраны: скорость работы алгоритма, наиболее оптимальное найденное значение, среднее отклонение найденных решений. На основе проведенных исследований можно сделать вывод, что BA является высоко эффективным. В случае овражной или выпуклой целевой функции рекомендуется использовать параллельные вычисления для повышения эффективности за счет расчета нескольких стай летучих мышей одновременно, либо использовать повторный запуск на лучших решениях.

Ключевые слова: алгоритм летучих мышей, оптимизация, эвристический алгоритм, роевая оптимизация, стохастическая оптимизация

BAT ALGORITHM (BA) IN THE PROBLEM OF GLOBAL UNCONSTRAINED OPTIMIZATION

Lagunova A.D.
Institute of Automation and Control Processes
Far Eastern Branch of Russian Academy of Sciences
Vladivostok, Russia
690041, Vladivostok, 5 Radio Street
e-mail: schimka_06@mail.ru

ABSTRACT

In this paper, a study of the algorithm of a Bat Algorithm (BA) was conducted using the global unconstrained optimization problem as an example. The influence on the efficiency of the

algorithm of such parameters as the population size and the number of iterations was studied. Also, the most optimal values of population size and the number of iterations for 2- and 3-dimensional functions were identified. To study the effectiveness and comparative analysis of the algorithm, a software module was developed, which also implemented such methods as: direct search method, grid method, GWO, parallel direct search method and parallel grid method. As evaluation criteria were chosen: the speed of the algorithm, the most optimal value found, the average deviation of the solutions found. On the basis of the conducted research, it can be concluded that the algorithm BA is highly effective. In the case of a ravine or convex objective function, it is recommended to use parallel computing to increase efficiency by calculating several packs of bats at the same time, or to use the restart on the best solutions.

Key words: bat algorithm (BA), optimization, heuristic algorithm, swarm optimization, stochastic optimization

Введение

Оптимизационная задача – это задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств. Методы оптимизации находят широкое применение в области экономики, техники и управления. Функции, описывающие задачу управления или проектируемую систему, как правило, имеют сложный нелинейный характер, что не позволяет получать оптимальное решение в аналитической форме с помощью классических методов вариационного и дифференциального исчисления. Высокая вычислительная трудоемкость решения оптимизационных задач со стохастическим критерием заставляет искать способы достаточно быстрого получения желаемых результатов [1, 5].

В последние годы активно развиваются эвристические и метаэвристические алгоритмы оптимизации - алгоритмы, включающие практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения задачи [3, 4, 7, 12, 21]. Правильность данных алгоритмов для всех возможных случаев не доказана, однако известно, что такие алгоритмы дают достаточно хорошее решение в большинстве случаев. Данные алгоритмы позволяют ускорить решение задачи в 100-1000 раз, что особенно важно в задачах с большим числом переменных. Кроме того, эвристические алгоритмы позволяют найти решение даже в тех случаях, когда точное решение не может быть найдено или его поиск имеет большую вычислительную сложность. В настоящее время наиболее известным представителем эвристических методов является роевый интеллект, описывающий коллективное поведение децентрализованной самоорганизующийся системы. На данный момент существует большое количество роевых алгоритмов, например: метод роя частиц, муравьиный алгоритм, алгоритм кукушки и пр. [7, 11, 13, 8]. Одним из последних разработанных алгоритмов этого типа является алгоритм поиска летучих мышей.

Цель данной работы – исследование эффективности и сравнительный анализ алгоритма поиска летучих мышей, выявление недостатков алгоритма для функций разного типа и предложение решений для их устранения.

Алгоритм летучих мышей

Алгоритм летучих мышей (Bat Algorithm, BA) – метаэвристический алгоритм, разработанный Янгом в 2010 году [20], имитирующий свойство эхолокации летучих мышей [16]. Этот алгоритм потенциально более мощный, чем алгоритм роя частиц и генетический алгоритм, а также гармонический поиск. Более того, гармонический поиск и алгоритм роя частиц являются особыми случаями алгоритма летучих мышей при

соответствующих упрощениях [20]. Алгоритм может показаться немного сложнее, чем большинство других алгоритмов роевого интеллекта, однако он может быть достаточно эффективно применен к проблемам оптимизации и давать хорошие результаты, затрачивая меньшее количество времени [15, 20].

Большинство видов летучих мышей обладает удивительно совершенными средствами эхолокации, которая используется ими для обнаружения добычи и препятствий, а также для обеспечения возможности разместиться в темноте на насесте [6]. Параметры лоцирующего звукового импульса летучих мышей различных видов меняются в широких пределах, отражая их различные охотничьи стратегии. Большинство мышей используют короткие частотно-модулированные в пределах примерно одной октавы сигналы. В тоже время некоторые виды не используют частотную модуляцию своих звуковых импульсов. В основу алгоритма летучих мышей положены следующие три правила.

1. С помощью эхолокации все мыши могут определять расстояние до добычи и препятствий, а также различать их.

2. Мыши движутся случайным образом. Текущее положение и скорость i-ой мыши b_i, где $i = 1, ..., N$, N – размерность популяции летучих мышей, равны $x_i = (x_1, ..., x_D)$ и $v_i = (v_1, ..., v_D)$ соответственно, D – размерность пространства поиска. Для поиска добычи мыши генерируют сигналы, имеющие частоту i_ω и громкость a_i. В процессе поиска мыши могут менять частоту этих сигналов, а также частоту повторения излучаемых импульсов $r \in [0;1]$;

3. Частота сигналов может изменяться в диапазоне $[\omega^{min}, \omega^{max}]$, $\omega^{max} > \omega^{min} \geq 0$, а громкость сигналов в пределах [0;1].

Предположим, что необходимо решить задачу глобальной безусловной минимизации. Тогда схема алгоритма летучих мышей выглядит следующим образом[2]:

- *Инициализация популяции* ***B*** *и соответствующих параметров случайным образом*
- *определение* ***GBest***
- ***пока*** *не выполнится критерий остановки*
  - *обновить координаты летучих мышей (осуществить миграцию)*
  - ***для*** *каждой летучей мыши* $\mathbf{b_i}$ *сгенерировать случайное число* ***rand*** $\in [0;1]$
    - ***если rand >*** $\mathbf{r_i}$
- *находим ее лучшее решение, и в его окрестности реализуем локальный поиск*
- *найденное решение принимаем за текущее положение мыши*
- *сгенерировать в окрестности* $\mathbf{b_i}$ *случайным образом новое решение*
- *сгенерировать случайное число* ***Rand*** *в пределах [0;1]*
- ***если Rand <*** $\boldsymbol{a_i}$ *и* $\boldsymbol{f(b_i) < f(GBest)}$
  - *принимаем это решение в качестве нового положения мыши*
  - *уменьшаем громкость сигнала*
  - *увеличиваем частоту повторения сигналов*
  - *конец цикла*
  - *обновить значение* ***GBest***
- *конец цикла*

В данной схеме *GBest* – это лучшее найденное решение всеми летучими мышами в популяции. На этапе инициализации алгоритма начальные значения частот ω_i^0, громкостей a_i^0 и частот повторения импульсов r_i^0, где $i = 1, ..., N$, полагаются равномерно распределенными в соответствующих интервалах $[\omega^{min}, \omega^{max}]$, $[a^{min}, a^{max}]$, $[0,1]$.

Миграция летучей мыши b_i, $i = 1, ..., N$, осуществляется по формулам [2,20]:

$$x_i' = x_i + v_i',$$
$$v_i' = v_i + \omega_i * (x_i - x^{best}),$$
$$\omega_i = \omega^{min} + (\omega^{max} - \omega^{min}) * R(0;1),$$

где x_i', x_i – новые и старые координаты *i*-ой летучей мыши, v_i', v_i – новое b старое значения скорости *i*-ой летучей мыши, *R*(0;1) – случайное число из интервала (0;1).

Другими словами, на каждой итерации летучая мышь перемещается в направлении, определяемом суммой вектора перемещения на предыдущей итерации и случайным образом возмущенного вектора направления на лучшую мышь $(x_i - x^{best})$. Рассмотренная процедура миграции алгоритма летучих мышей имеет много общего с аналогичной процедурой стайного алгоритма оптимизации.

Случайный локальный поиск выполняется по следующей схеме:

1. Случайным образом варьируется текущее положение *i*-ой летучей мыши b_i в соответствии с формулой:

$$x_i' = x_i + \bar{a} * R(-1,1),$$

где *i* = 1, ..., *N*, x_i', x_i – новые и старые координаты *i*-ой летучей мыши $\bar{a}$ –текущее среднее значение громкостей всех летучих мышей в популяции, такое что

$\bar{a} = \frac{1}{N} * \sum_{i=1}^{N} a_i$;

$R(-1,1)$ – величина, равномерно распределенная на интервале от -1 до 1;

2. Вычисляется значение целевой функции в новой точке: $f(x_i') = f_i'$. Если оно лучше предыдущего значения, то есть $f_i' < f_i$, то процедура локального поиска завершается, в противном случае фиксированное число раз осуществляется возврат к шагу 1.

Эволюция параметров a_i и r_i осуществляется по правилам:

$$a_i^{t+1} = b_a * a_i^t,$$
$$r_i^{t+1} = r_i^0 * (1 - \exp(-b_r * t)),$$

где *i* = 1, ..., *N*, a_i^{t+1}, a_i^t – громкости *i*-ой мыши на (*t* + 1)-ой и *t*-ой итерациях соответственно, r_i^0 – частота повторения импульсов *i*-ой мышью при инициализации, r_i^{t+1}– частота повторения импульсов *i*-ой мышью на (*t* + 1)-ой итерации, *t* – номер поколения (итерации), $b_a \in (0,1), b_r > 0$ – заданные константы (свободные параметры алгоритма), рекомендуемые значения которых равны 0.9 [2, 15, 20].

Исследование эффективности алгоритма

В ходе исследований, алгоритм был реализован в классическом виде для решения задачи оптимизации. Также была реализована возможность задать такие параметры работы алгоритма, как: максимальное количество итераций, размер популяции и количество варьируемых параметров для нахождения оптимального значения целевой функции.

Для теста были выбраны три функции (Рис.2) [13,17,19]:

- Функция Де-Джонга (2- и 3-мерная). Глобальный минимум равен 0 при $x_i = 0$:

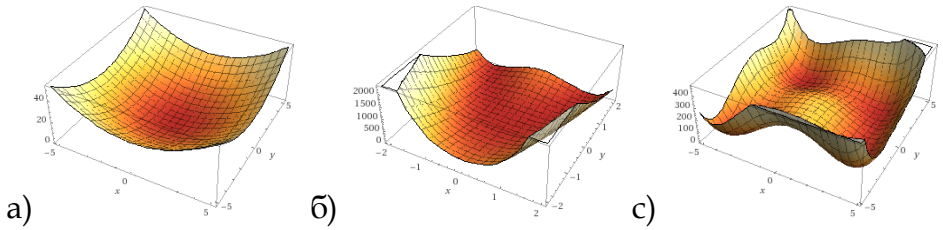
$$f(x) = \sum_{i=1}^{n} x_i^2, \quad x_i \in [-5.12; 5.12].$$

- Функция Розенброка (2- и 3-мерная). Глобальный минимум равен 0 при $x_i = 1$:

$$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right], \quad x_i \in [-2{,}048; 2{,}048].$$

- Функция Химмельблау (2-мерная). Глобальный минимум равен 0 при $f(3;\ 2),\ f(-2.805118;\ 3.131312),\ f(-3{,}779310;\ -3{,}283186),$ $f(3.584428;\ -1.848126)$:

$$f(x) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2, \quad x_i \in [-5; 5].$$



а) б) с)

Рис.2. *а) функция Де-Джонга, б) функция Розенброка, с) функция Химмельблау*

Алгоритм летучих мышей запускался 20 раз для каждого набора входных данных, чтобы оценить результаты в среднем. Для оценки эффективности алгоритма использовались такие параметры, как: наименьшее время работы алгоритма в секундах (*time*), наиболее оптимальное найденное значение во всей серии (*min*), среднее отклонение от реального оптимума (*S*). Результаты сравнивались с методом полного перебора, методом сеток, их параллельной реализацией (*N* от 2 до 8 процессов), а так же с методом стаи серых волков (GWO) [8 ,9, 13]. В таблицах приведены только лучшие по скорости выполнения параллельные алгоритмы для каждого набора данных. Для большей информативности даны также сведения о размере пространства поиска (количество точек на входе) и о количестве точек, для которых реально производились вычисления (*point*).

На основании приведенных ниже данных (таблицы 1-6), что алгоритм ВА высоко эффективен в задачах как с одноэкстремальной, так и с мультомодальной целевой функцией.

Функция Де-Джонга

Для выпуклой функции Де-Джонга алгоритм оказался достаточно эффективным (см. Таблица 1). Не смотря на довольно большое пространство поиска (более 1 млрд. точек), время работы эвристических алгоритмов занимает всего долю секунды, что, в случае двух переменных, почти в 4000 раз быстрее, чем метод полного перебора. В случае трех переменных, ВА работает более чем в 1 млн. раз быстрее. Метод сеток, также как и полный перебор, проигрывает ВА, по скорости, но по нахождению точного минимума и средней погрешности решений показывает более высокие результаты. Однако стоит отметить, что решения метода сеток зависят от удачного выбранного разбиения сетки. В данном случае обе сетки были выбраны так, чтобы попасть на заранее известный минимум. В общем же случае, даже при наложении более плотной сетки (например, 0.1) высока вероятность получить значение целевой функции довольно далекое от глобального оптимума.

По сравнению с алгоритмом стаи серых волков, ВА также показывает более высокую скорость, однако заметно уступает в вероятности нахождения точного минимума. Это объясняется тем, что в случае выпуклой функции стае проще найти и окружить 1 жертву, так как ее поведение координируется тремя альфами [13, 8], в то время как летучие мыши получают информацию только от одной мыши с лучшим решением. Более того, только мыши с высокой частотой излучаемых импульсов (~1-5% от общего количества) быстро приблизятся к лучшему решению. Подавляющее же большинство мышей довольно медленно будет приближаться к лучшему решению или же, наоборот,

будут удаляться от него в поисках другого оптимального варианта. Кроме того, некоторые мыши в виду недостаточной громкости сигналов могут «не услышать», что рядом с ними есть решение лучше, чем то, что у них есть на данный момент, а, следовательно, и не передвинуться в этом направлении. Такое поведение в некоторой степени негативно сказалось на производительности алгоритма в случае выпуклой функции. Визуализация поведения 20-ти летучих мышей при нахождении минимума выпуклой функции Де-Джонга за 20 итераций представлена на рисунке 1.

Таблица 1

Функция Де-Джонга

	2 переменные							3 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S	Кол-во точек на входе	iter	N	point	time	min	S
Полный перебор 0,01	1 048 576	-	1	1 048 576	11,916	**0**	-	1 073 741 824	-	1	1 073 741 824	15 755	**0**	-
		-	8		**2,814**	**0**	-		-	8		**3 867**	**0**	-
Сетка 0,16	4 096	-	1	4 096	0,111	**0**	-	262 144	-	1	262 144	3,913	**0**	-
		-	2		**0,106**	**0**	-		-	6		**1,009**	**0**	-
Сетка 0,32	1024	-	1	1024	**0,104**	**0**	-	32 768	-	1	32 768	0,605	**0**	-
		-	2		0,106	**0**	-		-	3		**0,205**	**0**	-
Стая 20 волков	1 048 576	10	1	200	**0,004**	0,0001	0,0261	1 073 741 824	10	1	200	**0,006**	0,0094	0,245
		20		400	0,008	**0**	0,0003		20		400	0,011	0,0002	0,0115
		30		600	0,011	**0**	**0**		30		600	0,018	**0**	0,0031
		40		800	0,015	**0**	**0**		40		800	0,022	**0**	**0,0001**
Стая 30 волков		10		300	**0,006**	0,0004	0,0029		10		300	**0,008**	0,0017	0,0809
		20		600	0,013	**0**	0,0001		20		600	0,017	0,0001	0,0044
		30		900	0,018	**0**	**0**		30		900	0,026	**0**	0,0004
		40		1200	0,022	**0**	**0**		40		1200	0,032	**0**	**0,00003**
20 летучих мышей	1 048 576	10	1	200	**0,001**	0,0105	0,5722	1 073 741 824	10	1	200	**0,001**	0,5187	2,3134
		20		400	0,001	0,0005	0,3689		20		400	0,002	0,1004	1,9987
		30		600	0,003	**0**	0,4676		30		600	0,003	0,0714	1,9038
		40		800	0,003	**0**	**0,3331**		40		800	0,004	**0,0159**	**1,7930**
30 летучих мышей		10		300	**0,001**	0,0138	0,3731		10		300	**0,002**	0,1141	1,1460
		20		600	0,003	**0**	0,3221		20		600	0,003	0,0940	1,1248
		30		900	0,004	**0**	0,2303		30		900	0,004	0,0001	1,2912
		40		1200	0,005	**0**	**0,1749**		40		1200	0,006	**0,0028**	**0,6905**

Среднее отклонение найденных решений алгоритмом ВА заметно проигрывает всем вышеуказанным методам, за исключением метода сеток, в виду сильной зависимости от правильного выбора сетки. Однако при анализе полученных данных прослеживается уменьшение среднего отклонения при увеличении количества итераций или популяции мышей. А поскольку скорость ВА во много раз выше алгоритма полного перебора и метода сеток, можно без особого ущерба во времени увеличить количество особей или итераций до получения желаемой точности (см. Таблица 2). В данном случае ВА

превосходит полный перебор и метод сеток по всем параметрам. Однако, даже показывая более высокую точность, BA работает на 0,01 сек. медленнее GWO.

Также, при анализе полученных данных было выявлено, что при увеличении размера популяции снижается не только количество расчетов, необходимых для решения со 100% вероятностью нахождения точного минимума, но также и время работы алгоритма. Если для популяции из 20 мышей требуется 10 тыс. вычислений, то для 100 мышей количество вычислений снижается до 8 тыс. Кроме того, были получены рекомендуемые значения для нахождения оптимума со 100% вероятностью (отмечены в Таблице 2)

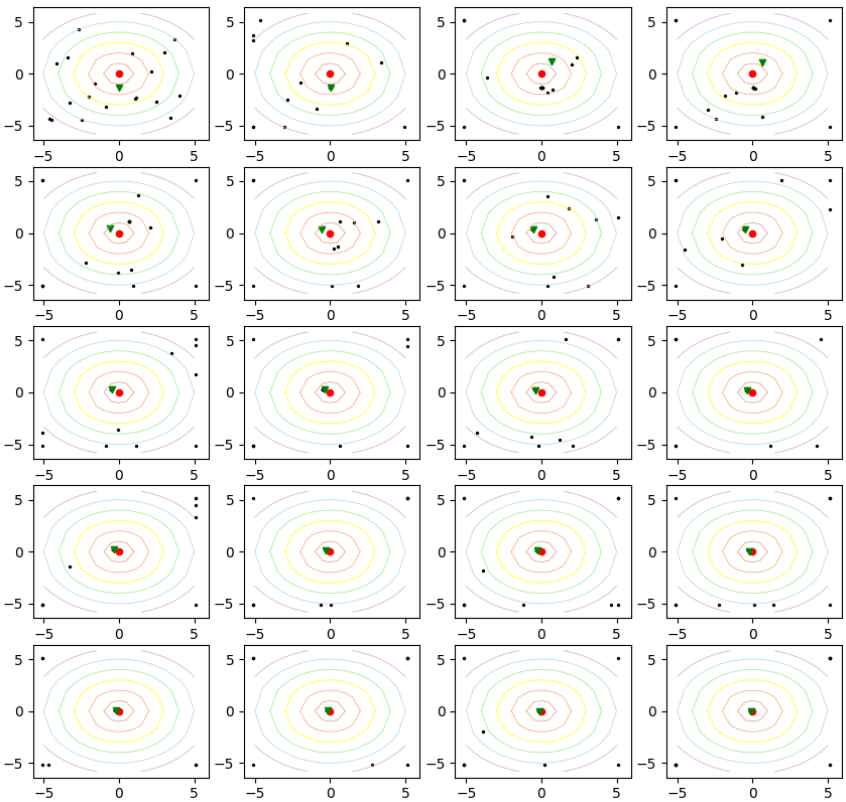


Рисунок 1. *Поведение стаи летучих мышей на выпуклой функции Де-Джонга*

Красная точка – реальный минимум; зеленый треугольник – лучшее решение, найденное мышами; черные точки – летучие мыши.

Таблица 2

Повышенные параметры BA для функции Дэ-Джонга

	2 переменные							3 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S	Кол-во точек на входе	iter	N	point	time	min	S
20 летучих мышей	1 048 576	100	1	2000	0,0081	0	0,2783	1 073 741 824	100	1	2000	0,0104	0,00001	1,1837
		400		8000	0,0325	0	0,0291		500		10000	0,0523	0	0,0089
		500		**10000**	**0,0404**	**0**	**0**		600		**12000**	**0,0630**	**0**	**0**
30 летучих мышей	1 048 576	80	1	2400	0,0099	0	0,0436	1 073 741 824	80	1	2400	0,0125	0	0,6893
		300		9000	0,0370	0	0,0043		200		6000	0,0309	0	0,0328
		350		**10500**	**0,0420**	**0**	**0**		300		**9000**	**0,0462**	**0**	**0**
50	1 048	20	1	1000	0,0040	0	0,0513	1 073 741	30	1	1500	0,0080	0	0,3826

летучих мышей	576	100		5000	0,0207	0	0,0006	824	100		5000	0,0260	0	0,0547
		180		**9000**	**0,0392**	**0**	**0**		180		**9000**	**0,0459**	**0**	**0**
100 летучих мышей	1 048 576	5	1	500	0,0023	0,0002	0,0465	1 073 741 824	20	1	2000	0,0107	0	0,1173
		10		1000	0,0045	0	0,0099		40		4000	0,0212	0	0,0001
		80		**8000**	**0,0340**	**0**	**0**		80		**8000**	**0,0413**	**0**	**0**

Функция Розенброка

На данной овражной функции алгоритм ВА, ожидаемо, работает эффективнее, чем остальные методы (см. Таблица 3). Показывая более высокую скорость среди всех методов, ВА находит решения намного ближе к глобальному оптимуму (в серии из 20 тестов).

Рассматривая подробнее поведение эвристических алгоритмов, можно заметить, что, в случае овражной функции, волки колеблются между тремя лучшими значениями, предполагая, что жертва между ними. Лишь некоторая часть волков будет искать другую жертву, что, в случае успеха, опять заставит стаю метаться между решениями [8]. У мышей поведение иное (см. Рисунок 2). Буквально на 1-2 итерации мыши находят одно решение в овраге, и, как уже говорилось выше, очень малый процент мышей на каждой итерации подлетает близко к лучшему решению. Основная же масса мышей не сильно ограничена в поисках, что позволяет уже на 2-5 итерации случайно найти еще более лучшее решение в овраге. На 4-20 итерации все больше мышей подлетают ближе к лучшему решению и производят поиск в малой окрестности, за счет чего происходит медленное, но точное движение вдоль оврага к жертве. Очевидно, при увеличении количества итераций, летучие мыши, продолжая движение вдоль оврага, обнаруживают точный минимум.

Таблица 3

Функция Розенброка

	2 переменные							3 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S	Кол-во точек на входе	iter	N	point	time	min	S
Полный перебор 0,001	16 777 216	-	1	16 777 216	255,8	0	-	68 719 476 736	-	1	68 719 476 736	~ 12 дней	0	-
		-	8		**59,12**	**0**	-		-	8		**~ 35 часов**	**0**	-
Сетка 0,01	167 281	-	1	167 281	2,7	0,0004	-	68 417 929	-		68 417 929	1682	0,0008	-
		-	8		**0,71**	**0,0004**	-		-			**405,5**	**0,0008**	-
Сетка 0,1	1 600	-	1	1 600	0,11	0,0797	-	64 000	-	1	64 000	1,807	0,1690	-
		-	4		**0,10**	**0,0797**	-		-	6		**0,409**	**0,1690**	-
Стая 20 волков	16 777 216	10	1	200	**0,003**	0,5700	16,91	68 719 476 736	10	1	200	**0,005**	0,7715	5,196
		20		400	0,008	0,0300	0,106		20	1	400	0,010	0,4075	1,911
		30		600	0,013	0,0081	0,071		30	1	600	0,017	0,4156	1,860
		40		800	0,017	**0,0025**	**0,027**		40	1	800	0,021	**0,3901**	**1,353**
Стая 30 волков	16 777 216	10	1	300	**0,006**	0,0413	0,158	68 719 476 736	10	1	300	**0,007**	0,4880	3,585
		20		600	0,011	0,0210	0,103		20	1	600	0,016	**0,1701**	1,505
		30		900	0,018	0,0041	0,096		30	1	900	0,024	0,6204	1,283
		40		1200	0,025	**0,0034**	**0,030**		40	1	1200	0,032	0,3711	**1,160**

20 летучих мышей	16 777 216	10	1	200	**0,001**	0,0085	0,4816	68 719 476 736	10	1	200	**0,002**	0,2887	8,4892
		20		400	0,002	0,0003	0,1937		20	1	400	0,003	0,1809	8,8357
		30		600	0,003	0,0001	0,2416		30	1	600	0,004	0,2075	**2,1023**
		40		800	0,004	**0,0003**	**0,1854**		40	1	800	0,0046	**0,0751**	6,5193
30 летучих мышей	16 777 216	10	1	300	**0,002**	0,0081	0,3824	68 719 476 736	10	1	300	**0,002**	0,0985	4,7685
		20		600	0,003	0,0043	0,5869		20	1	600	0,004	0,0984	1,9697
		30		900	0,004	0,0031	0,2682		30	1	900	0,005	0,0073	2,4781
		40		1200	0,007	**0,0015**	**0,2064**		40	1	1200	0,007	**0,0137**	**1,6260**

Результаты работы алгоритма ВА для овражной функции являются более чем удовлетворительными, однако 100% вероятность нахождения точного минимума так и не была достигнута. Имея явное преимущество в скорости, можно повысить точность за счет увеличения популяции и кол-ва итераций. Рекомендуемые значения для нахождения оптимума со 100% вероятностью отмечены в Таблице 4.

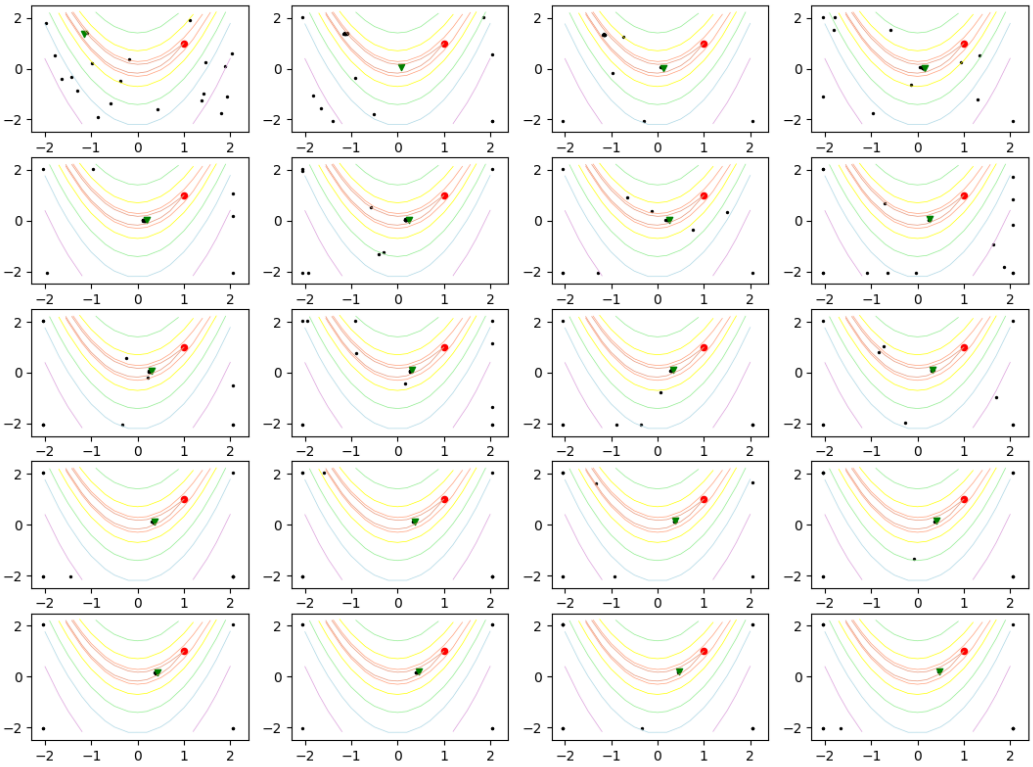


Рисунок 2. *Поведение стаи летучих мышей на овражной функции Розенброка*

Красная точка – реальный минимум; зеленый треугольник – лучшее решение, найденное мышами; черные точки – летучие мыши.

Таблица 4

Повышенные параметры ВА для функции Розенброка

	2 переменные							3 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S	Кол-во точек на входе	iter	N	point	time	min	S
20 летучих мышей	16 777 216	300	1	6000	0,0265	0	0,0204	68 719 476 736	900	1	18000	0,1022	0,000007	0,0115
		400		8000	0,0352	0	0,0013		1500		30000	0,1688	0	0,0021
		500		**10000**	**0,0441**	**0**	**0,00002**		3000		**60000**	**0,3466**	**0**	**0**
30 летучих мышей	16 777 216	200	1	6000	0,0262	0	0,0017	68 719 476 736	1000	1	30000	0,1742	0,000001	0,0009
		300		9000	0,0390	0	0,0007		1500		45000	0,2575	0	0,000004
		350		**10500**	**0,0454**	**0**	**0,00002**		2000		**60000**	**0,3508**	**0**	**0**
50 летучих мышей	16 777 216	80	1	4000	0,0175	0	0,0110	68 719 476 736	700	1	35000	0,2033	0,0000005	0,0009
		100		5000	0,0223	0	0,0047		1000		50000	0,2021	0	0,0000002
		200		**10000**	**0,0447**	**0**	**0,00009**		1200		**60000**	**0,3472**	**0**	**0**
100 летучих мышей	16 777 216	20	1	2000	0,0089	0	0,0133	68 719 476 736	300	1	30000	0,1713	0,0000001	0,0001
		80		8000	0,0357	0	0,0001		500		50000	0,2870	0	0,0000008
		100		**10000**	**0,0438**	**0**	**0**		600		**60000**	**0,3433**	**0**	**0**

Анализируя полученные данные, можно заметить, что, в отличие от функции Де-Джонга, увеличение популяции при том же количестве вычислений для функции Розенброка (2-мерной) не оказывает значительного влияния на точность и скорость алгоритма. Однако необходимое количество вычислений функции для отыскания точного минимума хотя бы раз в серии, напротив, напрямую зависит от размера популяции. Если для 20 мышей нужно как минимум 6 тыс. вычислений, чтобы точный минимум был найден хотя бы раз, то для 100 мышей достаточно 2 тыс. В трехмерном случае увеличение популяции при равном количестве вычислений незначительно уменьшает среднюю ошибку, не влияя на остальные показатели.

Функция Химмельблау

Для мультимодальной функции алгоритм ВА показал самые лучшие результаты по сравнению с другими алгоритмами (см. Таблица 5).

Таблица 5

Функция Химмельблау

	2 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S
Полный перебор 0,01	1 000 000	-	1	1 000 000	17,423	**0**	-
		-	8		**4,197**	**0**	-
Сетка 0,1	10 000	-	1	10 000	0,203	**0**	-
		-	4		**0,105**	**0**	-
Сетка 0,5	400	-	1	400	**0,104**	**0**	-
		-	2		0,105	**0**	-
Стая 20 волков	16 777 216	10	1	200	**0,004**	0,096	2,476
		20		400	0,008	0,150	1,305
		30		600	0,011	0,092	1,046

		40		800	0,018	**0,013**	**0,301**
Стая 30 волков	16 777 216	10	1	300	**0,007**	0,068	1,022
		20		600	0,011	0,013	0,816
		30		900	0,020	0,013	0,458
		40		1200	0,025	**0,006**	**0,395**
20 летучих мышей	16 777 216	10	1	200	**0,002**	0,953	3,478
		20		400	**0,002**	0,318	1,830
		30		600	0,003	**0,002**	1,571
		40		800	0,004	0,102	**0,605**
30 летучих мышей	16 777 216	10	1	300	**0,002**	0,0000008	1,984
		20		600	0,003	0,000001	1,703
		30		900	0,004	**0,0000001**	0,325
		40		1200	0,005	**0,0000001**	**0,143**

Поведение эвристических алгоритмов аналогично поведению на функции Розенброка. В GWO три альфы быстро «нападают на след» разных жертв, что заставляет стаю метаться между ними. В случае BA, летучие мыши, как уже было отмечсено, не так ограничены, что позволяет исследовать все пространство поиска, находя самые оптимальные варианты (см. Рисунок 3).

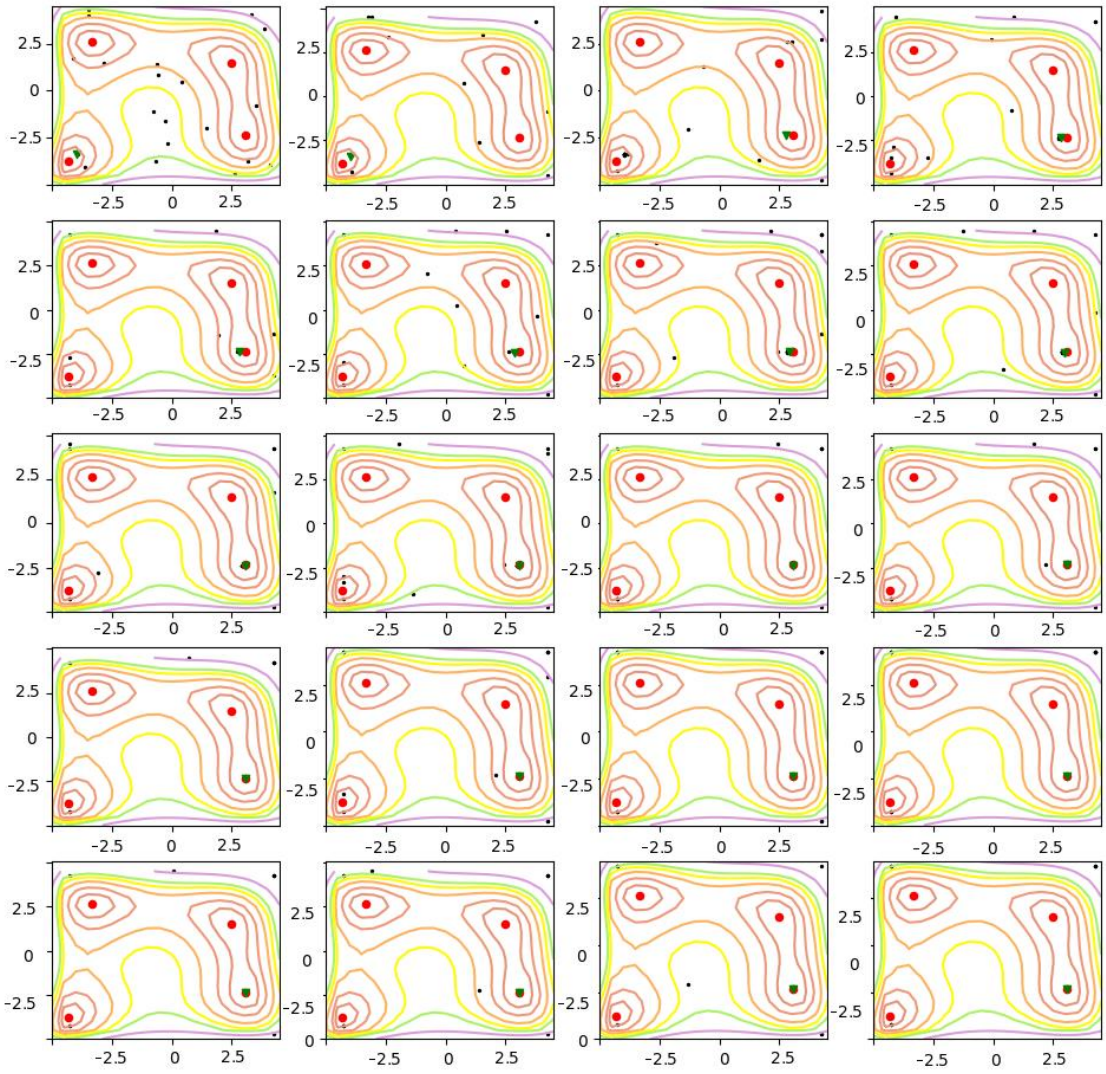


Рисунок 3. *Поведение летучих мышей на мультимодальной функции Химмельблау*

Красные точки – реальные минимумы; зеленый треугольник – лучшее решение, найденное мышами; черные точки – летучие мыши.

Для повышения точности увеличим количество итераций и размер популяции (см. Таблицу 6). Анализируя полученные данные, можно заметить, что в случае мультимодальной функции выгоднее увеличивать размер популяции, чем количество итераций, т.к. чем больше популяция, тем меньше затрачивается время на работу алгоритма, и тем меньше необходимо произвести расчетов для достижения 100% вероятности нахождения точного минимума. Например, при популяции из 20 мышей необходимо произвести 10 тыс. вычислений и 0,045 секунд, чтобы со 100% вероятностью можно было бы гарантировать нахождение точного минимума, в то время как для 100 мышей достаточно 4 тыс. вычислений и в 2.5 меньше времени. Рекомендуемые значения для нахождения оптимума со 100% вероятностью отмечены в Таблице 6.

Таблица 6

Повышенные параметры ВА для функции Химмельблау

	2 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S
20 летучих мышей	16 777 216	100	1	2000	0,009	0	0,576
		400		8000	0,035	0	0,000001
		500		**10000**	**0,045**	**0**	**0**
30 летучих мышей	16 777 216	80	1	2400	0,010	0	0,271
		200		6000	0,026	0	0,000001
		250		**7500**	**0,033**	**0**	**0**
50 летучих мышей	16 777 216	30	1	1500	0,005	0	0,088
		80		4000	0,017	0	0,000001
		100		**5000**	**0,022**	**0**	**0**
100 летучих мышей	16 777 216	5	1	500	0,003	0,0000002	0,328
		30		3000	0,014	0	0,0000001
		40		**4000**	**0,018**	**0**	**0**

Методы повышения эффективности

На основе приведенных исследований эффективности алгоритма ВА и его поведения в случае различный целевых функций, можно сделать вывод о высокой эффективности алгоритма для всех видов функций. Однако для достижения 100% вероятности нахождения точного минимума требуется значительное увеличение времени и количества расчетов целевой функции.

Для повышения эффективности алгоритма возможно использование не увеличение популяции стаи летучих мышей, а увеличение количества стай. При этом у каждой стаи должна быть своя территория, на которую запрещено залетать посторонним мышам. Каждой стае мышей придется обследовать меньшую территорию, что в случае выпуклой функции позволит тщательнее обследовать область, содержащую оптимум, а в овражных функциях не даст всем мышам скапливаться вместе и ускорит их продвижение вдоль оврага. В случае функции с несколькими экстремумами, деление на несколько стай так же может повысить точность нахождения минимума.

Чтобы смоделировать поведение нескольких стай летучих мышей можно разбить вычисления на несколько процессов, каждый из которых будет моделировать поведение одной стаи на выделенной ей территории. Многопроцессорные вычисления позволят не только с большей вероятностью находить точные экстремумы, но и не увеличивать время расчетов за счет параллельной обработки данных для каждой стаи.

Еще одним методом повышения эффективности может стать вторичный запуск мышей на места наиболее вероятного расположения жертвы. Это позволит мышам игнорировать особенности поверхности функций вне этой области и быстрее находить оптимум.

Заключение

Высокая вычислительная трудоемкость решения оптимизационных задач со стохастическим критерием заставляет искать способы достаточно быстрого получения желаемых результатов. На данный момент разработано не малое количество методов и алгоритмов, в известной мере ускоряющих процедуру поиска оптимальных решений. Наиболее радикальным направлением, сокращающим трудоемкость решения сложных вычислительных задач, стали эвристичекие алгоритмы, одним из которых является алгоритм летучих мышей (BA).

Сравнительный анализ алгоритма BA показал, что:

- алгоритм показывает достаточно высокую эффективность для овражных и мультимодальных функций
- алгоритм является более эффективным, по сравнению с методом сеток, особенно, на большом пространстве поиска
- алгоритм BA эффективнее GWO на овражных и мультимодальных функциях
- на выпуклых функциях BA имеет высокую погрешность
- увеличение количества итераций или популяции позволяет в определенной степени повысить точность вычислений и снизить погрешность
- для уменьшения погрешности алгоритма BA целесообразней увеличивать популяцию, а не количество итераций
- увеличение популяции не только повышает точность вычислений, но и уменьшает их количество, приводя к сокращению времени работы алгоритма
- повышение точности за счет количества итераций ведет к значительному увеличению количества вычислений целевой функции, что подходит не для всех задач
- использование параллельных вычислений для запуска алгоритма с несколькими стаями мышей позволит эффективнее находить экстремума в случае овражных или выпуклых функций, не увеличивая время работы
- для повышения точности алгоритма целесообразен повторный запуск мышей в окрестности найденных решений

Список литературы

1. Абрамов О.В., Катуева Я.В. Технология параллельных вычислений в задачах анализа и оптимизации / О.В. Абрамов, Я.В. Катуева // Проблемы управления. – 2003. №4. – С. 11-15
2. Ахмедова, Ш.А. Коллективный самонастраивающийся метод оптимизации на основе бионических алгоритмов [Текст]: дис. канд. тех. наук: 05.13.01 / Ш.А. Ахмедова. - Красноярск, 2016. - 150 с.
3. Ахмедова Ш.А. Последовательный и параллельный стайный алгоритм для задач условной и безусловной оптимизации / Ш.А. Ахмедова // Актуальные проблемы авиации и космонавтики. – 2012. – Т.1. №8. – С. 289-290

4. Брестер К.Ю. О решении задач многокритериальной оптимизации самонастраивающимся генетическим алгоритмом / К.Ю. Брестер // Актуальные проблемы авиации и космонавтики. – 2012. – Т.1. №8. – С. 290-291
5. Диго Г.Б., Диго Н.Б., Катуева Я.В. Применение детерминированных критериев в задачах стохастической оптимизации / Г.Б. Диго, Н.Б. Диго, Я.В. Катуева // Многопроцессорные вычислительные системы. – 2006. - №2(12). - С. 82-88
6. Карпенко А. П. Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов // А.П. Карпенко // Приложение к журналу «Информационные технологии». – 2012. - №7. - С.1-32.
7. Кулиев Э.В., Щеглов С.Н., Пантелюк Е.А., Кулиева Н.В. Адаптивный алгоритм стаи серых волков для решения задач проектирования / Э.В. Кулиев, С.Н. Щеглов, Е.А. Пантелюк, Н.В. Кулиева // Известия ЮФУ. Технические науки. - 2017 . - №7. - С.28-38.
8. Лагунова, А.Д. Алгоритм стаи серых волков (GWO) для задач оптимизации / А.Д. Лагунова // Оригинальные исследования (ОРИС). – 2019. – №4. – С.52-62
9. Лагунова А.Д., Назаров Д.А. Параллельный алгоритм решения задачи оптимального параметрического синтеза на основе метода сеток / А.Д. Лагунова, Д.А. Назаров // Труды Международного симпозиума "Надежность и качество". - 2018. - Т.1. - С. 255-258
10. Матренин П.В. Методы стохастической оптимизации [Текст]: учебное пособие / П.В. Матренин, М.Г. Гриф, В.Г. Секаев. – Новосибирск: НГТУ, 2016. - 65 с.
11. Орловская Н.М. Анализ биоинспирированных методов глобальной оптимизации // Труды МАИ. – 2014. - №73. URL: https://mai.ru/upload/iblock/85d/85d945530545d38e6d4cbd591417766d.pdf (дата обращения: 17.04.2019)
12. Сагун А.В., Хайдуров В.В., Кунченко-Харченко В.И. Метод стаи волков и его модификация для решения задачи поиска оптимального пути / А.В. Сагун, В.В. Хайдуров, В.И. Кунченко-Харченко // Фізико-математична освіта: науковий журнал. – 2017. - №2(12). – С. 135-139
13. Сергиенко А. Б. Тестовые функции для глобальной оптимизации / А.Б. Сергиенко. – Красноярск: Изд-во СГАУ им. М.Ф. Решетнева, 2015. – 112 с.
14. Частикова В.А., Дружинина М.А., Кекало А.С. Исследование эффективности алгоритма поиска косяков рыб в задаче глобальной оптимизации // Современные проблемы науки и образования. - 2014. - №4. - URL: http://science-education.ru/ru/article/view?id=14142 (дата обращения: 17.04.2019)
15. Частикова. В.А., Новикова Е.Ф. Алгоритм летучих мышей для решения задачи глобальной оптимизации // Научные труды КубГТУ. – 2015. - №2 - URL: https://ntk.kubstu.ru/file/348 (дата обращения: 17.06.2019)
16. Altringham, J.D. Bats: Biology and Behaviour. Oxford University Press, 1996.
17. Bossek Y. SMOOF: Single- and Multi-Objective Optimization Test Functions / Y. Bossek // The R Journal. - 2017. - Vol. 9 (1). - P. 103-113
18. Mirjalili S., Lewis A. Grey Wolf Optimizer / S. Mirjalili, A. Lewis // Advanced in Engineering Software. - 2014. - Vol 69. – P. 46-61.
19. Molga M., Smutnicki C. Test functions for optimization needs: [Электронный ресурс]. 2005. URL: https://www.vafaeijahan.com/en/wp-content/uploads/2012/02/Test-functions-for-optimization-needs.pdf (Дата обращения: 17.04.2019).
20. Yang X.S. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization / Xin-She Yang // Springer, SCI 284. – 2010. - P. 65-74
21. Zong W.G., Williams J.C. Ecological Optimization using Harmony Search / W.G. Zong, J.C. Williams // American conference on applied mathematics. – 2008. – P. 148-152

References

1. Abramov OV, Katueva Ya.V. Technology of parallel computing in the tasks of analysis and optimization / O.V. Abramov, Ya.V. Katueva // Management problems. - 2003. №4. - P. 11-15 [in Russian].
2. Akhmedova, Sh.A. Collective self-adjusting optimization method based on bionic algorithms [Text]: PhD dissertation: 05.13.01 / Sh.A. Akhmedova. - Krasnoyarsk, 2016. – P. 150 [in Russian].
3. Akhmedova Sh.A. Consecutive and parallel sharing algorithm for problems of conditional and unconditional optimization / Sh.A. Akhmedova // Actual problems of aviation and cosmonautics. - 2012 - T.1. №8. - P. 289-290 [in Russian].
4. Brester K.Yu. On solving problems of multicriteria optimization of a self-adjusting genetic algorithm / K.Yu. Brester // Actual problems of aviation and astronautics. - 2012 - T.1. №8. - P. 290-291 [in Russian].
5. Digo G. B, Digo N. B, Katueva Ya.V. Application of deterministic criteria in problems of stochastic optimization / G. B. Digo, N.B. Digo, Ya.V. Katueva // Multiprocessor Computing Systems. - 2006. - №2 (12). P. 82-88 [in Russian].
6. Karpenko A.P. Population Algorithms for Global Search Engine Optimization. Overview of new and little-known algorithms // A.P. Karpenko // Appendix to the magazine "Information Technology". - 2012. - №7. - P.1-32 [in Russian].
7. Kuliev E.V., Shcheglov S.N., Pantelyuk E.A., Kuliyeva N.V. Adaptive algorithm for solving design problems / E.V. Kuliev, S.N. Scheglov, E.A. Pantelyuk, N.V. Kulieva // News of SFU. Technical science. - 2017 - №7. - P.28-38 [in Russian].
8. Lagunova, A.D. Algorithm of a pack of gray wolves (GWO) for optimization problems / A.D. Lagunova // Original Studies (ORIS). - 2019. - №4. - P.52-62 [in Russian].
9. Lagunova A.D., Nazarov D.A. Parallel algorithm for solving the problem of optimal parametric synthesis based on grid parameters / A.D. Lagunova, D.A. Nazarov // Proceedings of the International Symposium "Reliability and Quality." - 2018. - T.1. - P. 255-258 [in Russian].
10. Matrenin P.V. Methods of stochastic optimization [Text]: study guide / P.V. Matrenin, M.G. Grief, V.G. Sekaev. - Novosibirsk: NSTU, 2016. – 65 p. [in Russian].
11. Orlovskaya N.M. Analysis of bioinspired methods of global optimization // Proceedings of the MAI. - 2014. - №73. URL: https://mai.ru/upload/iblock/85d/85d945530545d38e6d4cbd591417766d.pdf (access date: 17.04.2019) [in Russian].
12. Sagun A.V., Khaidurov V.V., Kunchenko-Kharchenko V.I. The method of searching for optimal paths / A.V. Sagun, V.V. Khaidurov, V.I. Kunchenko-Kharchenko // Physics and Mathematics Journal of Science. - 2017. - №2 (12). - P. 135-139 [in Russian].
13. Sergienko A. B. Test functions for global optimization / A. B. Sergienko. - Krasnoyarsk: Publishing House of SSAU them. Mf Reshetnev, 2015. – 112 p. [in Russian].
14. Chastikova V.A., Druzhinina M.A., Kekalo A.S. Investigation of the effectiveness of the algorithm for searching fish schools // Modern problems of science and education. - 2014. - №4. - URL: http://science-education.ru/ru/article/view?id=14142 (access date: 17.04.2019) [in Russian].
15. Chastikova. V.A., Novikova E.F. The Bat Algorithm for solving the problem of global optimization // Scientific works of KubGTU. - 2015. - №2 - URL: https://ntk.kubstu.ru/file/348 (access date: 17.06.2019) [in Russian].
16. Altringham, J.D. Bats: Biology and Behaviour. Oxford University Press, 1996.
17. Bossek, Y. SMOOF: single- and multi-purpose test optimization functions / Yu. Bossek // R. Magazine - 2017. - Tom. 9 (1). - P. 103-113

18. Mirdzhalili S., Lewis A. Gray wolf Optimizer / S. Mirdzhali, A. Lewis // Advanced level in engineering software. - 2014. - Volume 69. - From 46-61.
19. Molga M., Smutnitsky S. Test functions for optimization needs: [Electronic resource]. 2005. URL: https://www.vafaeijahan.com/en/wp-content/uploads/2012/02/Test-functions-for-optimization-needs.pdf (Revised: 04/17/2019).
20. Yang X.S. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization / Xin-She Yang // Springer, SCI 284. – 2010. - P. 65-74
21. Zong W.G., Williams J.C. Environmental Optimization Using Harmony Search / W.G. Zong, J.C. Williams // American Conference on Applied Mathematics. - 2008. – P. 148-152