УДК 681.5.015.63-192

© 2021 г. **О.В. Абрамов**, д-р техн. наук,

А.Д. Лагунова

(Институт автоматизации и процессов управления ДВО РАН, Владивосток)

ОБ ИСПОЛЬЗОВАНИИ ЭВРИСТИЧЕСКИХ АЛГОРИТМОВ В ЗАДАЧАХ ОПТИМАЛЬНОГО ПАРАМЕТРИЧЕСКОГО СИНТЕЗА

Рассматривается проблема оптимального синтеза технических систем с учетом закономерностей случайных производственных и эксплуатационных изменений их параметров. Для решения возникающих при этом задач предложены некоторые пути и средства создания эффективных эвристических алгоритмов многомерной поисковой оптимизации. Основное внимание уделено алгоритму летучих мышей и созданию его параллельной модификации.

Ключевые слова: параметрический синтез, параметр, техническая система, случайный процесс, эвристический алгоритм, оптимизация, метод, технология параллельных вычислений.

DOI: 10.22250/isu.2021.68.34-46

Введение

Проблема оптимального параметрического синтеза (ОПС) технических устройств и систем с учетом стохастических закономерностей отклонений их параметров от расчетных и требований надежности связана с необходимостью решения задач высокой вычислительной сложности. Суть ОПС состоит в поиске таких начальных (номинальных) значений параметров элементов системы (внутренних параметров), при которых обеспечивается максимальная вероятность выполнения условий работоспособности в течение заданного времени эксплуатации. При этом предполагается, что структура (топология) проектируемой системы и ее математическая модель известны [1, 2]. Функции, описывающие проектируемую систему, обычно имеют сложный нелинейный характер, что не позволяет получить оптимальное ре-

шение в аналитической форме с помощью классических методов дифференциального и вариационного исчисления. Поэтому для решения задачи ОПС необходимо использовать алгоритмические поисковые методы. Стохастический характер критерия оптимальности, многомерность пространства поиска, необходимость решения задачи глобальной оптимизации заставляют искать пути создания эффективных численных методов решения задач ОПС. Одним из наиболее радикальных путей решения задач высокой вычислительной сложности является использование технологии параллельных (распределенных) вычислений [2, 3]. Однако большинство методов поисковой оптимизации имеют принципиально последовательный характер. Свойством потенциального параллелизма обладает разве что метод сканирования (полного перебора) [4, 5]. Вместе с тем в последнее время все большее внимание привлекают эвристические методы оптимизации, которые в большинстве своем поддаются распараллеливанию. Идея создания эффективных эвристических алгоритмов поисковой оптимизации, ориентированных на решение задач ОПС, рассматривается в данной работе. Основное внимание уделяется алгоритму летучих мышей и созданию его параллельной модификации. Проведен анализ алгоритма летучих мышей, а также исследовано влияние увеличения количества стай, популяции и итераций на примере задачи выбора номинальных значений параметров электронной схемы. Для реализации алгоритма был разработан программный модуль на языке Python.

Эвристический роевой алгоритм летучих мышей

Роевой интеллект, описывающий коллективное поведение децентрализованной самоорганизующейся системы, является наиболее известным представителем эвристических методов. На данный момент существует большое количество роевых алгоритмов – , например, метод роя частиц, алгоритм стаи серых волков, алгоритм кукушки, муравьиный алгоритм и пр. [6].

Одним из последних разработанных поисковых алгоритмов этого типа является алгоритм летучих мышей. Алгоритм летучих мышей (Bat Algorithm, BA) – метаэвристический алгоритм, разработанный Янгом (Xin-She Yang) в 2010 г. [7], имитирующий свойство эхолокации летучих мышей [8]. Являясь несколько более сложным, чем большинство других роевых алгоритмов, он обеспечивает достаточно высокую эффективность при меньших временных затратах.

В основу алгоритма BA положены следующие правила:

- 1) благодаря эхолокации мыши определяют расстояние до добычи;
- 2) передвижение мышей хаотично. Текущее положение и скорость i -й

мышь равны соответственно $x_i = (x_1, \dots, x_D)$ и $v_i = (v_1, \dots, v_D)$, $i = 1, \dots, N$; N – размерность популяции; D – размерность пространства поиска. Для поиска добычи мышь генерирует сигналы с частотой i_ω и громкостью a_i . В процессе поиска мышь может менять частоту сигнала, а также частоту повторения излучаемых импульсов $r \in [0;1]$;

3) частота сигналов изменяется в диапазоне $[w^{min}, w^{max}]$ $w^{max} > w^{min} \geq 0$, а громкость сигналов – в пределах $[0;1]$.

На этапе инициализации алгоритма начальные значения частот ω_i^0 , громкостей a_i^0 и частот повторения импульсов r_i^0 , $i = 1, \dots, N$ полагаются равномерно распределенными в соответствующих интервалах $[w^{min}, w^{max}]$, $[a^{min}, a^{max}]$, $[0, 1]$. Миграция летучей мыши b_i , $i = 1, \dots, N$ описывается следующими соотношениями [7, 9]:

$$x_i' = x_i + v_i', \quad (1)$$

$$v_i' = v_i + \omega_i * (x_i - x^{best}), \quad (2)$$

$$\omega_i = \omega^{min} + (\omega^{max} - \omega^{min}) * R(0;1), \quad (3)$$

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

Случайный локальный поиск выполняется по следующей схеме:

1. Случайным образом варьируется текущее положение i -й летучей мыши b_i :

$$x_i' = x_i + \bar{a} * R(-1,1), \quad (4)$$

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

$$\bar{a} = \frac{1}{N} * \sum_{i=1}^N a_i; \quad (5)$$

$R(-1, 1)$ – 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, т.е. $f_i' < f_i$, 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:

$$a_i^{t+1} = b_a * a_i^t, \quad (6)$$

$$r_i^{t+1} = r_i^0 * (1 - \exp(-b_r * t)), \quad (7)$$

где a_i^{t+1} , a_i^t – громкости i -й мыши соответственно на $(t+1)$ -й и t -й итерациях, $i = 1, \dots, N$; r_i^0 – частота повторения импульсов i -й мышью при инициализации; r_i^{t+1} – частота повторения импульсов i -й мышью на $(t+1)$ -й итерации; t –

номер поколения (итерации); $b_a (0, 1)$, $b_r > 0$ – заданные константы (свободные параметры), значения которых равны 0.9 [7, 9].

Параллельная реализация алгоритма летучих мышей

Алгоритм ВА достаточно эффективен для задач оптимизации, однако средняя погрешность на некоторых функциях может оказаться достаточно большой [10]. Для ее уменьшения требуется значительное увеличение времени и количества расчетов целевой функции, что, в свою очередь, не всегда гарантирует высокой вероятности нахождения точного экстремума.

Одним из вариантов решения данной проблемы может стать увеличение количества стай, а не количества мышей внутри одной стаи. Каждой стае в таком случае должна быть выделена своя территория, на которую доступ другим мышам будет запрещен. Такое разделение позволит каждой стае точнее обследовать выделенную территорию и повысит вероятность нахождения экстремума. При такой реализации алгоритма овражные и мультимодальные функции в некоторой степени должны перестать быть проблемными. В случае функции с несколькими экстремумами деление на несколько стай должно позволить каждой стае более тщательно исследовать свою территорию и не отвлекаться на другие решения вне ее границ, что, в свою очередь, позволит увеличить вероятность нахождения нескольких экстремумов.

Чтобы смоделировать такое поведение, можно разбить вычисления на несколько процессов, каждый из которых будет моделировать поведение одной стаи на выделенной ей территории. Пример такого разделения для функции Химмельблау показан на рис. 1. Параллельные вычисления могут позволить не только с большей вероятностью находить точные экстремумы, но и не дать значительного увеличения времени расчетов за счет параллельной обработки данных для каждой стаи.

Рис. 1. Разделение пространства поиска для функции Химмельблау:

а) 1 процесс; б) 2 процесса; в) 3 процесса; г) 4 процесса.

Для удобства обозначим $ВА(N)$ – алгоритм летучих мышей, запущенный для N – различных стай, каждая из которых выполняет поиск на отдельной «территории», и результаты поиска обрабатываются в отдельном процессе.

Для реализации параллельных вычислений можно использовать библиотеку MPI (mpi4py) языка Python.

В общем виде реализуется следующий алгоритм:

задается число процессов N и начальные данные (целевая функция, границы области поиска, количество итераций, количество летучих мышей в стае, начальные параметры алгоритма);

нулевой процесс делит область поиска на n равных частей и передает их всем процессам (в том числе себе);

все процессы выполняют алгоритм ВА на заданных интервалах, после чего передают результаты в нулевой процесс;

нулевой процесс обрабатывает полученные данные и выдает результат.

Исследование эффективности алгоритма

В ходе исследований алгоритм ВА был реализован в классическом виде. Также была реализована возможность задать такие параметры работы алгоритма как: максимальное количество итераций, размер популяции, количество варьируемых параметров для нахождения оптимального значения целевой функции и их области допустимости, ограничения на входные характеристики. Для оценки эффективности алгоритма использовались: наименьшее время работы алгоритма в секундах ($time$); наибольшие найденные значения целевой функции (запаса работоспособности) – $F(W^*)$. Значение целевой функции вычисляется с точностью $1 \cdot 10^{-7}$.

Для проверки эффективности алгоритма решим задачу выбора номинальных значений параметров схемы транзисторно-транзисторной логики [11]. Пусть требуется рассчитать оптимальные значения параметров резисторов R_1 , R_2 , R_3 , R_4 (варьируемые параметры) в ИС ТТЛ, принципиальная схема которой изображена на рис. 2.

Рис.2. Схема ТТЛ.

Единицы измерения, интервалы допуска на варьируемые параметры и выходные характеристики даны в табл. 1, 2.

Таблица 1

Варьируемые параметры				
Обозначение параметра	R1, кОм	R2, кОм	R3, кОм	R4, кОм
Интервал допуска	[0.50; 10]	[0.50; 10]	[0.05; 1]	[0.10; 2]

Таблица 2

Условия работоспособности		
Обозначение параметра	Интервал допуска	Описание параметра
N	[20; ...]	Коэффициент нагружения
$\Delta U_{ном}, В$	[0.8; ...]	Уровень помехи в логическом состоянии «0»
$P, мВт$	[0; 20]	Средняя потребляемая схемой мощность
S	[1.5; ...]	Степень насыщения транзистора Т ₂
$t_{зрс}, нс$	[0; 15]	Задержка распространения сигнала

Условия работоспособности связаны с параметрами элементов схемы следующими соотношениями [12]:

$$N = 2.1 \left\{ 2.45 + R_1 \left[\frac{3.95}{R_2 + 0.011} - \frac{1}{R_4} \right] \right\}, \quad (8)$$

$$\Delta U_{ном} = 0.9 - 0.03 \ln(R_4), \quad (9)$$

$$P = \frac{19}{R_1} + \frac{10.8}{R_2 + 0.017}, \quad (10)$$

$$S = 8.55 \frac{R_2}{R_1}, \quad (11)$$

$$t_{зд,р} =$$

$$t_{зд,р} = 0.55 + 0.084 \frac{R_2}{R_3} + \sqrt{R_2^2 (6 - 0.079 N)^2 + \left(0.69 - 0.0024 N \frac{R_2}{R_1} \right) (5.7 + 0.15 N) R_1 R_2 -} \quad (12)$$

$$- 0.5 R_2 (3 + 0.079 N) - 0.5 * 8 \frac{1 - 0.134 R_2 - 0.136 \frac{R_2}{R_4} - 0.026 N \frac{R_2}{R_1}}{1 - 0.134 R_2 + \frac{6.9}{8 + 3 R_3}} -$$

$$- 0.5 R_2 (4.9 + 0.055 N).$$

Целевую функцию будем искать согласно предложенным в [12] формулам. При оптимизации схемы нужно найти такой вектор $\mathbf{W}^*$, который обеспечивает максимум минимального запаса работоспособности схемы в

допустимой области пространства управляемых параметров, т.е.

$$F(W^*) = \max \min z_j(W), \quad j \in [1; m], \quad (13)$$

где F – целевая функция; W – вектор управляемых параметров; z_j – оценка запаса работоспособности.

Оценку запаса работоспособности целесообразно нормировать с учетом технологического разброса значений параметров [13]:

$$z_j = a_j [(TT_j - y_{jnom}) / \delta_j - 1], \quad (14)$$

где δ_j – величина, характеризующая разброс j -го параметра; a_j – весовой коэффициент; TT_j – граница поля допуска для j -го входного параметра; y_{jnom} – вычисленное фактическое значение j -го входного параметра.

В частном случае нормального распределения параметра y_j под δ_j можно понимать утроенное среднее квадратичное отклонение. Тогда оценка z_j равна количеству трехсигмовых допусков на j -й выходной параметр, укладывающийся в интервал $[y_{jnom}, TT_j]$. Значения δ_j взяты равными 20% от TT_j [13]. Весовой коэффициент a_j определяет степень важности параметра и его приоритет при распределении запаса работоспособности. Так, наиболее важные параметры относят к первой группе, а наименее важные – ко второй. Значение коэффициента для параметров первой группы всегда равно 1. Для параметров второй группы значение выбирается в интервале $[5; 50]$, при этом чем больше значение a_j , тем меньший запас работоспособности получит j -й параметр. Согласно [11], все входные параметры, кроме S , принадлежат первой группе. Действительно, транзистор T_2 обязательно должен входить в насыщение, но заметное перевыполнение условия работоспособности параметра S нецелесообразно, вследствие чего ему была дана оценка $a_j=10$.

Поскольку мы не знаем вид целевой функции, воспользуемся формулой для функции общего вида [10], согласно которой оптимальное количество популяции и итераций равны 504 и 50-60 соответственно. Для параметров $p = 500$ и $i = 50$ алгоритм летучих мышей нашел решение:

$$F(W^*) = 0,643802,$$

$$t = 0,613 \text{ сек.}$$

Сравним полученные результаты с часто используемой в системах автоматизированного проектирования (САПР) программой расчета параметров компонентов (РПК) [12] и параллельным алгоритмом построения области работоспособности (ОР) с помощью регулярной сетки [16]. Как видно из табл. 3, вычисленный минимальный запас работоспособности оказался в 4,5 раза больше, чем в случае использования алгоритма РПК, и в 4,2 раза больше, чем при использовании алгоритма регулярной сетки.

Таблица 3

Вычисленные значения параметров и целевой функции										
	$F(W^*)$	R1	R2	R3	R4	N	$\Delta U_{\text{пом}}$	P	S	$t_{\text{зд.р}}$
ВА	0.64	4.33	1.17	0.51	1.51	29.5	0.89	13.5	2.3	10.1
РПК	0.14	4.49	0.95	0.05	0.48	24.6	0.92	15.4	1.8	11.6
Регулярная сетка (парал.)	0.15	5.93	1.5	0.52	1.63	30.1	0.89	10.3	2.2	12.4

Наилучшие значения параметров, заданных интервалами допуска, должны находиться как можно ближе к середине этих интервалов. Если же задана только верхняя или нижняя граница интервала, то оптимальное значение должно находиться как можно дальше от границы. Из рис. 3 и 4 видно, что алгоритм ВА наравне с алгоритмом построения ОР с помощью регулярной сетки справился с этой задачей намного лучше, чем РПК.

Рис. 3. Вычисленные значения варьируемых параметров:

◇ ВА; ■ РПК; △ Регулярная сетка.

Рис. 4. Вычисленные значения входных параметров:

◇ ВА; ■ РПК; △ Регулярная сетка.

Тем не менее, получив достаточно хороший запас для большинства критериев работоспособности, предложенные в [12] и [15] алгоритмы дали слишком малый запас для критерия t (рис. 5). Однако совмещение метода расчета запаса работоспособности из [12] с оптимизационным алгоритмом ВА позволило найти решение с бóльшим запасом работоспособности, ускорив при этом время расчетов и повысив точность вычислений, вплоть до 15 знаков после запятой.

Рис.5. Нормированные запасы работоспособности входных параметров.

Теперь проверим работу алгоритма на нескольких процессах. Как видно из табл. 4, при достаточно малом увеличении времени (0,08 сек.) алгоритм ВА (4) нашел решение с несколько большим запасом работоспособности.

Таблица 4

Результаты работы ВА(N)						
Proc	F(W*)	t	R1	R2	R3	R4
1	0.643802	0.613	4.32716623	1.174503798	0.511241967	1.513005852
2	0.652304	0.634	4,68522125	1,139909364	0,684754272	1,240600549
3	0.653595	0.646	5.07469957	1.103154175	0.579454512	1.716802148
4	0.655993	0.696	4.97090808	1.113442985	0.803494557	1.054287178

Отметим, что небольшой прирост запаса работоспособности обусловлен размерностью задачи: ведь чем больше количество варьируемых параметров, тем сильнее выражено увеличение эффективности распараллеливания.

Если же в приоритете стоит не увеличение точности, а уменьшение времени расчетов без существенной потери точности, то можно уменьшить количество итераций или сократить популяцию.

Согласно данным, представленным в табл.5, благодаря распараллеливанию алгоритма можно сократить время в 4,8 раза при незначительной потере точности.

Таблица 5

Результаты работы ВА(4) с различными значениями популяции и итераций							
Pop	iter	F(W*)	t	R1	R2	R3	R4
500	30	0.65123	0.417	4.59278	1.14955	0.69306	1.63943
	10	0.65411	0.187	4.90677	1.11848	0.54322	1.46383
400	50	0.64627	0.569	4.68944	1.13650	0.50002	0.93351
	30	0.63308	0.352	4.29369	1.17742	0.54554	1.26440
	10	0.65361	0.127	5.00773	1.10900	0.51906	1.11504

Pop	iter	F(W*)	t	R1	R2	R3	R4
200	50	0.64693	0.286	4.71165	1.13447	0.41438	1.86559
	30	0.65002	0.190	5.03910	1.10456	0.70172	0.57246
	10	0.65592	0.097	4.92243	1.11788	0.72427	1.11593
100	50	0.63599	0.166	4.51601	1.15072	0.97990	0.69174
	30	0.62805	0.100	4.13317	1.19775	0.71954	1.81718
	10	0.63405	0.030	4.28442	1.17913	0.68795	1.20971
50	50	0,57942	0,094	4,55241	1,11886	0,19292	0,51854
	30	0,53065	0,076	3,73819	1,21101	0,19087	1,10511
	10	0,62033	0,033	4,15940	1,18967	0,93169	0,94777
20	50	0,48265	0,028	4,47041	1,08274	0,09176	0,49679
	30	0,46470	0,034	3,74893	1,17333	0,13992	0,71757
	10	0,32472	0,006	4,49237	1,01435	0,05731	0,47036

Более того, сокращение времени даже в 100 раз уменьшило точность всего в 2 раза, что все равно позволяет получить запас работоспособности почти в 2,2 раза больше, чем в расчетах, представленных в работах [12] и [15, 16].

В дополнение к вышеизложенным данным стоит также отметить, что результаты серии тестов для разного количества процессов, итераций и популяций позволяют достаточно точно предположить, что функция имеет выпуклый вид: точность вычислений сохраняется при понижении заданных параметров вплоть до значений, принадлежащих выпуклым функциям [14].

Некоторые возможности повышения эффективности параллельного алгоритма летучих мышей

Рассмотрим более подробно вопрос деления территории для стай. В данной работе деление происходило по первому варьируемому параметру, в результате каждая стая получала достаточно узкую прямоугольную область.

Причем, чем больше стай, тем более вытянутой становится область поиска, что на разных типах функций будет затруднять поиск [17].

Для повышения эффективности будем делить территорию в несколько этапов (рис. 6).

1. Нулевой процесс запускает стаю №1 на всю территорию с использованием небольшого количества итераций. Решение этой стаи – точка. Далее нулевой процесс выделяет в пространстве поиска окрестность возле найденного решения.

2. Первый процесс получает координаты этой области и запускает на

нее стаю №2 (также используя малое число итераций). В это же время нулевой процесс продолжает поиск в своей области. Когда решение будет найдено на первом процессе, вокруг него также выделяется окрестность. Координаты передаются второму процессу для запуска следующей стаи. Вокруг второго решения нулевого процесса строится окрестность для третьего процесса.

3. Теперь 4 стаи ищут решение в своей области еще раз.

4. После завершения расчетов все данные снова соберутся в нулевом процессе.

Предложенный метод (scale-BA) позволяет строить квадратные, а не вытянутые области поиска, что должно благоприятно влиять на поиск.

Рис. 6. Деление территории (scale-BA).

Заключение

Решение оптимизационных задач в современных системах автоматизированного проектирования (САПР) связано с высокой вычислительной трудоемкостью. Данный факт вынуждает искать эффективные средства получения желаемых результатов. Наиболее радикальным направлением, сокращающим трудоемкость решения подобных вычислительных задач, могут стать эвристические алгоритмы и технология параллельных вычислений.

Для повышения эффективности эвристического алгоритма летучих мышей был реализован параллельный алгоритм BA(N), соединяющий в себе оба этих направления.

Анализ алгоритма BA(N) показал, что:

использование параллельных вычислений и деления пространства поиска на несколько территорий для разных стай повышает эффективность поиска экстремума целевой функции;

использование BA(N) позволяет достаточно точно предположить вид функции в случае серии тестов с разным количеством итераций и популяций;

эффективность алгоритма зависит от «удачно» выбранных территорий; для повышения эффективности алгоритма целесообразно использовать деление пространства поиска методом scale-BA.

ЛИТЕРАТУРА

1. *Абрамов О.В.* Методы и алгоритмы параметрического синтеза стохастических систем // Проблемы управления. – 2006. – № 4. – С. 3-8.
2. *Абрамов О.В.* Параллельные алгоритмы расчета и оптимизации надежности по постепенным отказам // Автоматика и телемеханика. – 2010. – № 7. – С. 126-135.
3. *Абрамов О.В., Катуева Я.В.* Параллельные алгоритмы анализа и оптимизации параметрической надежности // Надежность. – 2005. – №4. – С. 19-26.
4. *Абрамов О.В.* Проектирование технических систем с элементами настройки // Надежность и качество сложных систем. – 2014. – № 2(6). – С. 51-55.
5. *Лагунова А.Д., Назаров Д.А.* Параллельный алгоритм решения задачи оптимального параметрического синтеза на основе метода сеток // Труды Международного симпозиума "Надежность и качество". – 2018. – Т.1. – С. 255-258.
6. *Карпенко А. П.* Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов // Приложение к журналу «Информационные технологии». – 2012. – №7. – С.1-32.
7. *Yang X.S.* A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization // Springer, SCI 284. – 2010. – P. 65-74.
8. *Altringham, J.D.* Bats: Biology and Behaviour. Oxford University Press, 1996.
9. *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>.
10. *Лагунова А.Д.* Алгоритм летучих мышей (BA) для задачи безусловной оптимизации // Оригинальные исследования (ОРИС). – 2019. – №6. – С.101-116.
11. *Антушев Г.С.* Методы параметрического синтеза технических систем. – М.: Наука, 1989.
12. *Анисимов Б.В., Белов Б.И., Норенков И.П.* Машинный расчет элементов ЭВМ. – М.: Высшая школа, 1976.
13. Микроэлектроника. Сборник статей под ред. Ф.В. Лукина. Вып. 3. – М.: Советское радио, 1969.
14. *Лагунова А.Д.* Выбор оптимальных параметров алгоритма BA для задач безусловной оптимизации // Оригинальные исследования (ОРИС). – 2020. – Т. 10, №6. – С. 83 – 95.
15. *Назаров Д.А.* Разработка алгоритмических и программных средств построения и анализа областей работоспособности аналоговых технических систем: Дис... канд. техн. наук.– Владивосток: ИАПУ ДВО, 2011.
16. *Катуева Я.В., Назаров Д.А.* Аппроксимация и построение областей работоспособности в задаче параметрического синтеза // Труды Международного симпозиума «Надежность и качество». – Пенза: ПГУ, 2005. – Т.1. – С. 130-134.
17. *Лагунова А.Д.* Параллельные вычисления как способ повышения эффективности алгоритма летучих мышей (BA) // Сборник статей IX Международной научно-

практической конференции «Инновационное развитие науки и образования», 2020. – С.78-87.

E-mail:

Абрамов Олег Васильевич – abramov@iasp.dvo.ru;

Лагунова Алина Дмитриевна – schimka_06@mail.ru.