УДК 519.8 + 519.7

Разработка метода метаэвристического программирования для синтеза нелинейных моделей*

О.Г. Монахов, Э.А. Монахова

Институт вычислительной математики и математической геофизики Сибирского отделения Российской академии наук, просп. Акад. Лаврентьева, 6, Новосибирск, 630090

E-mails: monakhov@rav.sscc.ru (Монахов О.Г.), monakhov,emilia@rav.sscc.ru (Монахова Э.А.)

Английская версия этой статьи печатается в журнале "Numerical Analysis and Applications" № 4, Vol. 13, 2020.

Монахов О.Г., Монахова Э.А. Разработка метода метаэвристического программирования для синтеза нелинейных моделей // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2020. — Т. 23, № 4. — С. 415–429.

Рассматривается решение проблемы построения нелинейных моделей (математических выражений, функций, алгоритмов, программ) на основе заданных экспериментальных данных, множества переменных, базовых функций и операций. Разработан метод метаэвристического программирования для синтеза нелинейных моделей, который использует представление хромосомы в виде вектора действительных чисел и позволяет применить различные биоинспирированные (природоподобные) алгоритмы оптимизации при поиске моделей. Получены оценки эффективности предложенного подхода с использованием десяти различных биоинспирированных алгоритмов (генетический алгоритм — две модификации, алгоритм дифференциальной эволюции, алгоритм оптимизации роем частиц, алгоритм колонии пчел, алгоритм оптимизации на основе преподавания и обучения и его две модификации, эволюционная стратегия с адаптацией матрицы ковариаций, алгоритм поиска на основе теплопередачи) и проведено его сравнение со стандартным алгоритмом генетического программирования, алгоритмом грамматической эволюции и алгоритмом декартового генетического программирования. Проведенные эксперименты показали существенное преимущество предложенного подхода по сравнению с указанными алгоритмами как по времени поиска решения (более чем на порядок в большинстве случаев), так и по вероятности нахождения заданной функции (модели) (во многих случаях более чем в два раза).

DOI: 10.15372/SJNM20200405

Ключевые слова: *метод метаэвристического программирования, генетический алгоритм, генетическое программирование, алгоритм грамматической эволюции, декартово генетическое программирование, нелинейные модели, биоинспирированные алгоритмы, метаэвристические алгоритмы.*

Monakhov O.G., Monakhova E.A. Development of a metaheuristic programming method for the nonlinear models synthesis // Siberian J. Num. Math. / Sib. Branch of Russ. Acad. of Sci. — Novosibirsk, 2020. — Vol. 23, № 4. — P. 415–429.

The solution of the problem of building nonlinear models (mathematical expressions, functions, algorithms, programs) based on an experimental data set, a set of variables, a set of basic functions and operations is considered. A metaheuristic programming method for the evolutionary synthesis of nonlinear models has been developed that has a representation of a chromosome in the form of a vector of real numbers and allows the use of various bioinspired (nature-inspired) optimization algorithms in the search for models. The effectiveness of the proposed algorithm is estimated using ten bioinspired algorithms and compared with a standard algorithm of genetic programming, grammatical evolution and Cartesian Genetic Programming. The experiments have shown a significant advantage of this approach as compared with the above algorithms both with respect to time for the solution search (greater than by an order of magnitude in most cases), and the probability of finding a given function (a model) (in many cases at a twofold rate).

*Работа выполнена в рамках проекта РАН (проект № 0315-2016-0006).

Keywords: *metaheuristic programming method, genetic algorithm, genetic programming, grammatical evolution, Cartesian Genetic Programming, nonlinear models, bioinspired algorithms.*

1. Введение и основные определения

Рассматривается решение проблемы построения нелинейных моделей, представленных в виде математических выражений, функций, формул, алгоритмов, программ, на основе заданных экспериментальных данных, множества переменных, базовых функций и операций. Задачей является поиск математического выражения f^*, наилучшим образом описывающего нелинейную вычислительную модель, заданную совокупностью входных X и выходных Y экспериментальных данных, т. е. требуется подобрать такую функцию $Y = f^*(X)$, которая отображает зависимость Y от X с минимальной погрешностью (иногда эту задачу называют символьной регрессией или идентификацией системы). Поиск выражения f осуществляется на основе заданного множества базовых функций и операций $F_1 = \{f_i \mid f_i : R \times \cdots \times R \to R\}$ и множества переменных и констант $T_1 = \{x_i, c_i\}$, путем суперпозиции которых автоматически создаются аналитические выражения (формулы) f, представляющие модель, и компьютерные программы для их вычисления. Примем, что критерием качества модели является целевая функция FF (fitness function, функция качества, функция пригодности), которая вычисляет сумму квадратов отклонений выходных данных выражения $Y_i' = f(X_i)$ от заданных эталонных значений Y_i для определенных наборов множества входных данных выражения X_i: $FF = \sum_{i=1}^{N}(f(X_i) - Y_i)^2$, $1 \le i \le N$, где N — число экспериментальных данных. Целью алгоритма эволюционного синтеза нелинейных моделей является поиск выражения f для минимума целевой функции:

$$\min_{f \in D(F_1, T_1)} FF(f), \tag{1}$$

где $D(F_1, T_1)$ — множество всех моделей, образованных с помощью заданных множеств базовых функций и переменных.

Отметим актуальность данной задачи, связанной с проблемой построения систем автоматизации научных открытий, являющейся задачей большого вызова в области искусственного интеллекта (см., например, [1]). Одной из основных задач, возникающих при решении данной проблемы, является задача автоматического синтеза (открытия) описаний нелинейных математических моделей природных процессов и технических объектов на основе экспериментальных данных. Особую актуальность данная задача приобретает в настоящее время, когда в процессе научных исследований экспериментальные установки автоматически генерируют большие объемы данных для усвоения которых, анализа взаимосвязей переменных и построения теоретических моделей требуется развитие новых методов искусственного интеллекта, эволюционного поиска и оптимизации.

Одним из подходов к решению данной задачи является генетическое программирование (ГП) [2–5], которое представляет собой одно из направлений в эволюционном моделировании [6] и ориентировано в основном на решение задач автоматического синтеза программ на основе обучающих данных путем эволюционного поиска моделей, минимизирующих погрешность отображения входных данных в выходные. Хромосомы, представленные в виде дерева, автоматически генерируются с помощью генетических операторов в ГП и являются (после интерпретации) выражениями и реализующими их компьютерными программами различной величины и сложности.

В данной работе представлен новый метод метаэвристического программирования для эволюционного синтеза (МПЭС) нелинейных моделей, имеющий более высокую эффективность эволюционного поиска по сравнению с известными ранее системами генетического программирования. Предложенный алгоритм основан на использовании:

(1) представления генотипа нелинейных моделей в виде простого вектора действительных чисел (а не представление в виде более сложных структур — деревьев, сетей или программ, традиционных для генетического программирования),

(2) новых алгоритмов для преобразования этих векторов (генотипа) в фенотип (выражения и программы для представления нелинейной модели),

(3) организации эволюционного процесса поиска на множестве этих векторов с помощью обычного метаэвристического, биоинспирированного алгоритма оптимизации [7], с простыми эволюционными операторами поиска (а не специализированными операторами для более сложных структур),

(4) многовариантного метода кодирования нескольких решений в одном генотипе.

Проведенные эксперименты с использованием десяти различных биоинспирированных алгоритмов (генетический алгоритм — две модификации, алгоритм дифференциальной эволюции, алгоритм оптимизации роем частиц, алгоритм колонии пчел, алгоритм оптимизации на основе преподавания и обучения и его две модификации, эволюционная стратегия с адаптацией матрицы ковариаций, алгоритм поиска на основе теплопередачи) показали существенное преимущество предложенного подхода по сравнению с известными ранее алгоритмами как по времени поиска решения (более чем на порядок в большинстве случаев), так и по вероятности нахождения заданной функции (модели) (во многих случаях более чем в два раза).

Отметим, что новый метод МПЭС является дальнейшим развитием и обобщением предложенного ранее авторами метода многовариантного эволюционного синтеза (МВЭС) [8]. Он более универсален и унифицирован для использования с множеством метаэвристических (био- и природно-инспирированных) алгоритмов, а не применяет только один дискретный генетический алгоритм (ГА), как метод МВЭС, использует хромосомы в виде вектора действительных чисел, а не в виде вектора целых чисел как в методе МВЭС и, соответственно, использует другие операции декодирования хромосомы.

В следующей пункте будут приведены варианты представления решений (хромосом) в четырех наиболее известных системах генетического программирования. Далее описан новый алгоритм метаэвристического программирования для эволюционного синтеза нелинейных моделей, и в конце приведены экспериментальные оценки эффективности предложенного метода и рассмотренных ранее алгоритмов и проведено их сравнение.

2. Представление решений в системах генетического программирования

Рассмотрим варианты представления решений (хромосомы) в четырех наиболее известных системах генетического программирования: стандартное генетическое программирование (ГП, genetic programming) [2, 3], грамматическая эволюция (ГЭ, grammatical evolution) [9, 10], декартово генетическое программирование (ДГР, cartesian genetic programming) [11, 12], метод сетевого оператора (МСО) [13].

Поиск решения в процессе эволюции осуществляется на основе заданного множества базовых (элементарных) функций (function set), например $F_1 = \{+, -, /, *, \sin, \exp\}$ и т. д.

и заданного множества свободных проблемно-ориентированных переменных и констант (терминальное множество — terminal set, например $T_1 = \{x, y, a1, 2, 3.14\}$ и т. д.), из которых строится с помощью эволюционных операторов мутации, кроссовера, селекции требуемое математическое выражение (модель, программа).

Генетическое программирование. При стандартном генетическом программировании решение представляется в виде дерева вместо строки (вектора) чисел (целых, действительных, двоичных), используемых в генетическом алгоритме [6, 14]. При этом функции из набора базовых (элементарных) функций становятся внутренними узлами дерева решения, а элементы терминального множества (переменные и константы) становятся листьями (концевыми вершинами) дерева.

Операция кроссовера в этом случае состоит в порождении двух новых особей путем обмена частями хромосом родителей (обмен случайно выбранными поддеревьями для древовидной структуры хромосомы). Операция мутации при этом состоит в изменении значения случайно выбранной вершины в представлении функции f на другую случайно выбранную величину из множества допустимых значений. Отметим, что при древовидном представлении решения в ходе процесса эволюции возникает эффект неоправданного роста выражений и появления функционально лишних частей (интронов), что ведет к снижению эффективности эволюционного поиска [4].

Алгоритм грамматической эволюции. Классический алгоритм грамматической эволюции представляет собой комбинацию генетического алгоритма и заданной контекстно-свободной грамматики. В принципе, он использует отображение генотип–фенотип для интерпретации строки кодонов (векторов целых чисел, каждое из которых в интервале [0, 255]) для определения выражения на основе правил (продукций) контекстно-свободной грамматики, заданной в форме Бэкуса–Наура [9]. При декодировании генотипа используется простая формула для определения P — номера правила интерпретации каждого следующего кодона, если текущий символ X, то $P = K \bmod R$, где K — кодон, R — количество правил в заданной грамматике для символа X, X — некоторый нетерминальный символ.

При эволюции популяции векторов целых чисел (хромосом) в ГЭ используются операторы стандартного генетического алгоритма, а при их декодировании генотипа в фенотип (для определения функции пригодности) по указанному выше правилу получаются грамматически правильные функциональные выражения (программы). Теоретически этот механизм отображения работает для эволюции программ на любом языке. Детальное описание алгоритмической структуры ГЭ приведено в [9].

Декартово генетическое программирование. Декартово генетическое программирование является формой генетического программирования, которая использует хромосому фиксированного размера, представленную последовательностью целых чисел (генотип), кодирующую двумерную функциональную сеть (фенотип). Хромосома представляет в закодированном виде многокаскадную сеть из узлов, каждый узел которой состоит из базовой функции и набора связей с узлами предыдущих каскадов — входных переменных функции в узле. Первый каскад представляет входные переменные из терминального множества, а последний каскад представляет выходные переменные всей сети. Пути, соединяющие входные и выходные узлы и проходящие через ряд функциональных узлов, определяют аналитические выражения, состоящие из функций и входных переменных, принадлежащих данным путям.

Хромосома, определяющая функциональную сеть, состоит из трех типов генов (векторов целых чисел): функциональные гены, описывающие функции в узлах, гены связи, описывающие соединения между узлами, и выходные гены, описывающие выходные уз-

лы. Одна функциональная сеть может описывать несколько аналитических выражений, соответствующих разным выходным узлам. Вычисление функции пригодности происходит на всей сети для каждого набора входных данных. Полученные выходные значения сравниваются с требуемыми значениями, и отбираются сети с минимальными рассогласованиями этих значений. В ДГП эволюционный процесс наиболее часто организуется с использованием эволюционной стратегии типа (1 + 4)-ES [11] путем генерации случайных хромосом, оценки их пригодности, их мутации, как правило, с использованием алгоритма точечной мутации и отборе кандидатов для создания следующего поколения.

Метод сетевого оператора. Метод сетевого оператора является формой генетического программирования, при которой используется представление данного математического выражения в виде ориентированного графа без циклов — сетевого оператора. При этом сетевой оператор — ориентированный граф — состоит из узлов-источников (для представления входных переменных и параметров), узлов-стоков (для представления выходных переменных), промежуточных узлов (для представления бинарных операций) и ребер (дуг) графа (для представления унарных операций, включая тождественную унарную операцию). Сетевой оператор обычно представлен матрицей целых чисел, для более компактного представления сетевой оператор иногда представляют в виде списка или в виде "вектора сетевого оператора". В МСО эволюционный процесс организуется с использованием генетического алгоритма, при котором происходит поиск множества структурно-параметрических преобразований (вариаций) заданного в виде сетевого оператора базисного решения с целью получения оптимального математического выражения.

В следующем пункте будет описан новый алгоритм метаэвристического программирования для эволюционного синтеза нелинейных моделей, который, в отличие от описанных выше систем генетического программирования, имеет следующие особенности:

(1) простое представление хромосомы в виде вектора действительных чисел,

(2) каждая хромосома содержит в закодированном виде множество математических выражений в пространстве решений задачи,

(3) простые операции для декодирования хромосом,

(4) позволяет использовать множество метаэвристических алгоритмов для поиска оптимального решения.

3. Алгоритм метаэвристического программирования для эволюционного синтеза

Алгоритм метаэвристического программирования для эволюционного синтеза основан на моделировании процесса естественного отбора в популяции особей, каждая из которых представляет несколько точек в пространстве решений задачи оптимизации, а не единственное решение как в ГП, ГЭ, ДГП. Особи — хромосомы — представляют в закодированном виде математические выражения (формулы, программы) и являются последовательностями действительных чисел (векторами). Далее будем обозначать этот алгоритм как МПЭС$_r$. Каждая популяция является множеством хромосом, и каждая хромосома в данном алгоритме определяет множество выражений (формул), образующихся из нее после декодирования.

Алгоритм МПЭС состоит в эволюционном преобразовании множества хромосом (формул) в процессе естественного отбора с целью выживания "сильнейшей" особи, т. е.

особи с наименьшем значением целевой функции. Алгоритм начинается с генерации начальной популяции, особи которой создаются случайно, затем отбираются наилучшие особи путем декодирования генотипа (хромосомы) в фенотип (выражение) и вычисления функции пригодности. Для создания популяции следующего поколения новые особи формируются с помощью операций селекции (отбора), мутации и кроссовера (или других операций поиска, лежащих в основе используемого биоинспирированного алгоритма). Целью алгоритма является поиск минимума целевой функции FF (1). На практике, если получалось несколько решений с одинаковым значением целевой функции, то выбиралось решение с меньшей оценкой структурной сложности, т. е. с меньшей общей длиной (суммой числа элементов) формул решения.

Общая схема алгоритма метаэвристического программирования для эволюционного синтеза состоит в следующем:

1. Создание первоначальной популяции из случайно сгенерированных решений — векторов действительных чисел (хромосом). Решение представлено в хромосоме в закодированном виде — в виде генотипа.
2. Оценка популяции по заданной фитнес-функции (fitness function), которая показывает, насколько хорошо каждый индивид решает заданную проблему. При этом происходит декодирование генотипа в фенотип для интерпретации хромосомы как программы для вычисления фитнес-функции.
3. Создание популяции следующего поколения с помощью эволюционных операторов используемого биоинспирированного алгоритма (например, мутации, кроссовера, селекции).
4. Повторение п. 2 и п. 3, пока решение по заданному критерию не будет найдено или не будет достигнуто максимальное число поколений.

Приведенные базовые этапы моделирования процесса эволюции при использовании популяционных биоинспирированных алгоритмов не исключают применение различных стратегий организации популяции и других эволюционных операторов.

В данной работе предложен подход, основанный на декодировании основной структуры данных — хромосомы (вектора действительных чисел) в последовательность трехадресных команд (инструкций) для вычисления выражений (формул), закодированных в хромосоме. Для этого данная последовательность действительных чисел (хромосома) разбивается на группы из трех членов — триплеты (h_1, h_2, h_3), а каждая такая группа интерпретируется как трехадресная инструкция следующим образом: $<$ oper $><$ adr1 $><$ adr2 $>$, где операция oper выполняется над операндами, расположенными в инструкциях с номерами adr1 и adr2, которые вычисляются по следующим формулам, в этом случае хромосома МПЭС$_r$ — вектор действительных чисел с $0 < h_i < 1$:

$$\begin{gathered} \text{oper} = \lfloor h_1 * |F| \rfloor, \\ \text{adr1} = \lfloor (h_2 * (I-1)) \rfloor + 1, \\ \text{adr2} = \lfloor (h_3 * (I-1)) \rfloor + 1, \\ if\ \text{oper} = 0\ then\ \text{adr1} = \lfloor (h_2 * |T|) \rfloor + 1, \end{gathered} \tag{2}$$

где $|F|$ — мощность множества базовых функций, oper — номер элемента в этом множестве, т. е. номер исполняемой функции в текущей инструкции, I — номер текущей инструкции, adr1, adr2 — номера предыдущих инструкций, результаты исполнения которых используются как операнды в текущей инструкции, $|T|$ — мощность терминаль-

ного множества (если oper=0, то инструкция интерпретируется как оператор загрузки и происходит загрузка терминального символа с номером adr1).

В результате описанной процедуры декодирования хромосомы (вектора действительных чисел) получаем представление математического выражения интерпретируемым кодом, каждая команда (инструкция) которого будет считаться отдельной функцией (включающей все предыдущие команды). Первой операцией всегда будет команда вызова переменной данной функции (математического выражения). Выполнение происходит сверху вниз и аргументами команд могут выступать лишь команды, расположенные ранее, с меньшими номерами, поэтому исполнение получается линейным. Решение в этом случае представляет сразу множество функций-выражений (последовательностями команд от первого до каждого текущего оператора), что позволяет, в отличие от стандартного ГП, оценить одновременно множество выражений, представленных одной операторной последовательностью и сократить время поиска оптимального решения. При этом в качестве оценки данной хромосомы из множества полученных вариантов выбирается оценка выражения с минимальным значением целевой функции.

Переменные и константы искомой формулы f образуют множество терминальных символов T, а операции, используемые в формуле f, образуют множество нетерминальных символов F. Рассмотрим пример: пусть $f = (x+2)/e^{ax-5}$. При $F = \{L, +, -, /, *, \exp, \sin\}$ и $T = \{x, a, 2, 5\}$ имеем $|F| = 7$ и $|T| = 4$ (L — операция загрузки (вызова) переменной или константы из множества T с номером, вычисленном по первому адресу). Пример представления данного выражения f в виде последовательной операторной структуры с интерпретируемым кодом и в символьном виде показан в таблице 1, где I — номер инструкции, K — хромосома $(0.11, 0.14, \ldots, 0.91)$, разбитая на триплеты, M — результат декодирования триплета с помощью операций (2), CM — сама инструкция в мнемонической записи, E — получаемое выражение (формула) для данной инструкции в символьном виде, FC — значение выражения для каждой инструкции (при $x = 1,\ a = 2$). Приведем пример декодирования инструкции $I = 5$ по формулам (2): $K = 0.66{:}0.91{:}0.08$, номер операции oper $= \lfloor 0.66 * 7 \rfloor = 4$ — это операция умножения $*$ в F (здесь первый элемент L имеет номер 0), первый операнд adr1$= \lfloor 0.91 * (5-1) \rfloor + 1 = 4$, для инструкции с номером I_1=4 имеем $E_4 = a$ и $F_4 = 2$, второй операнд adr2 $= \lfloor 0.08 * (5-1) \rfloor + 1 = 1$, для инструкции с номером $I_2 = 1$ имеем $E_1 = x$ и $F_1 = 1$, отсюда: $M_5 = 4:4:1$, $CM_5 = *\ 4, 1$, $E_5 = E_4 * E_1 = ax$, $F_5 = F_4 * F_1 = 2$.

Таблица 1. Пример представления выражения в виде хромосомы (K), последовательной операторной структуры (CM) и в символьном виде (E)

I	K	M	CM	E	FC
1	0.11:0.14:0.97	0:1:–	$L\ x$	$E_1 = x$	$F_1 = 1$
2	0.13:0.55:0.15	0:3:–	$L\ 2$	$E_2 = 2$	$F_2 = 2$
3	0.19:0.16:0.79	1:1:2	$+\ 1, 2$	$E_3 = x + 2$	$F_3 = 3$
4	0.08:0.31:0.47	0:2:–	$L\ a$	$E_4 = a$	$F_4 = 2$
5	0.66:0.91:0.08	4:4:1	$*\ 4, 1$	$E_5 = ax$	$F_5 = 2$
6	0.04:0.79:0.81	0:4:–	$L\ 5$	$E_6 = 5$	$F_6 = 5$
7	0.35:0.80:0.87	2:5:6	$-\ 5, 6$	$E_7 = ax - 5$	$F_7 = -3$
8	0.77:0.96:0.19	5:7:–	$\exp\ \ 7$	$E_8 = e^{ax-5}$	$F_8 = 0.0498$
9	0.52:0.26:0.91	3:3:8	$/\ 3, 8$	$E_9 = (x+2)/e^{ax-5}$	$F_9 = 60.2566$

Отметим, что для данной хромосомы K определяется множество выражений E, вычисляется для каждого выражения значение целевой функции и из множества полученных оценок в качестве оценки данной хромосомы выбирается оценка выражения с минимальным значением целевой функции. Таким образом, в отличие от алгоритмов ГП, ГЭ и ДГП, которые кодируют одно решение в хромосоме, данный алгоритм кодирует несколько решений в хромосоме. Несколько решений в хромосоме кодируется также в MultiExpression Programming (MEP) [15] и в работе [16], но без использования унифицированного представления хромосом как простых векторов либо действительных, либо целых чисел и стандартных биоинспирированных алгоритмов оптимизации для их эволюции.

Отметим еще одно преимущество данного подхода: представление хромосомы в виде вектора действительных чисел и простого алгоритма его декодирования в нелинейную модель дает возможность использования этих векторов в рамках унифицированного (единого) подхода для поиска оптимальной модели с помощью различных метаэвристических популяционных алгоритмов, инспирированных природой [7]. В данной работе для поиска оптимальных моделей были использованы десять различных биоинспирированных алгоритмов: генетический алгоритм — две модификации — для векторов либо действительных чисел (обозначим такой алгоритм GA_r), либо целых чисел (GA_i) [8, 17], дифференциальной эволюции (DE) [18, 19], алгоритм оптимизации роем частиц (PSO) [20, 7], алгоритм колонии пчел (ABC) [21, 7], алгоритм оптимизации на основе преподавания и обучения (TLBO) [22] и его две модификации: (MTLBO) [23] и (SPM) [24], эволюционная стратегия с адаптацией матрицы ковариаций (CMA-ES) [25], алгоритм поиска на основе теплопередачи (SHTS) [26].

Рассмотрим в качестве примера использование алгоритма дифференциальной эволюции для организации эволюционного процесса популяции хромосом (векторов действительных чисел) в МПЭС. На шаге инициализации случайным образом генерируется популяция хромосом размера NP, состоящая из D-мерных векторов, каждый элемент которых находится в интервале $(0, 1)$ и число элементов кратно 3. Пусть эти начальные решения x_i^G, $i = 1, 2, \ldots, NP$, где G — номер поколения, охватывают все пространство поиска. При генерировании этих решений вычисляется значение функции пригодности в данных точках. Затем будем применять операторы мутации, кроссовера и селекции к данной популяции.

При операции мутации для каждого вектора x_i^G (целевой вектор) выбираются из популяции три случайных вектора: x_{r1}^G, x_{r2}^G, x_{r3}^G, такие что $i \neq r1 \neq r2 \neq r3$. Таким образом, начальная популяция должна быть по крайней мере размера четыре. С помощью целевого вектора и этих случайных векторов образуется новый вектор, известный как вектор мутации. Вектор мутации формируется таким образом, что взвешенная разность случайных векторов добавляется в целевой вектор. Мы используем следующее математическое уравнение (предложенное в [19]) для вычисления вектора мутации:

$$v_i^{G+1} = x_{r1}^G + K(x_{\mathrm{best}}^G - x_{r2}^G) + S_f(x_{\mathrm{best}}^G - x_{r3}^G).$$

Здесь v_i^{G+1} — вектор мутации, x_{r1}^G называется базовым вектором, вес S_f называется коэффициентом масштабирования или коэффициентом влияния, который лежит в пределах от 0 до 2, т. е. [0;2], и контролирует величину дифференциального изменения. Кроме коэффициента влияния S_f предлагаемая стратегия мутации имеет дополнительный параметр K, который называется коэффициентом направляющей силы. В то время как S_f имеет постоянное значение, K представляет собой динамический параметр, который изменяется случайным образом в каждом цикле и лежит в диапазоне [0;1]. Последние

два члена в правой части приведенного выше уравнения мутации позволяют учитывать разницу от лучшего вектора в популяции x^G_{best}. Это помогает в поддержании поиска в наилучшем направлении. Динамичный характер коэффициента направляющей силы K помогает в поддержании разнообразия, сохраняя при этом направление к лучшему вектору.

Оператор кроссовера применяется к целевому и мутантному векторам и образует пробный вектор. Пусть $u^{G+1}_{j,i} = \left(u^{G+1}_{1,i}, u^{G+1}_{2,i}, \ldots, u^{G+1}_{D,i}\right)$ — пробный вектор, он формируется следующим образом:

$$u^{G+1}_{j,i} = \begin{cases} v^{G+1}_{j,i}, & \text{если } R_{\text{rand}} \le C_r \ \forall j = j_{\text{rand}}, \\ x^G_{j,i} & \text{в противном случае,} \end{cases}$$

где C_r — вероятность кроссовера, R_{rand} — случайное число в интервале [0;1].

Оператор селекции реализует принцип выживания наиболее приспособленных индивидов. Он отбирает лучшие особи с минимальным значением фитнес-функции в текущей популяции путем сравнения между собой пробного и целевого векторов (вектор, для которого целевая функция минимальна, будет выбран для следующего поколения):

$$x^{G+1}_i = \begin{cases} u^{G+1}_i, & \text{если } FF(u^{G+1}_i) \le FF(x^G_i), \\ x^G_i & \text{в противном случае.} \end{cases}$$

Цикл операций мутации, кроссовера и отбора продолжается до достижения критерия остановки. Алгоритм завершается, когда найден вектор с $FF = 0$ или после заданного числа итераций (генераций) t. Отметим, что разработанный алгоритм метаэвристического программирования для эволюционного синтеза нелинейных моделей объединяет преимущества биоинспирированных алгоритмов оптимизации (простые операции над векторами, постоянный размер которых предотвращает эффект неоправданного роста выражений) и генетического программирования (автоматический синтез математических выражений и реализующих их компьютерных программ различной величины и сложности). Разработанный алгоритм совместно использует линейное представление хромосомы, простые операции (2) при декодировании генотипа в фенотип для интерпретации хромосомы как последовательности команд, многовариантный метод для представления множества моделей (выражений) с помощью одной хромосомы.

4. Экспериментальные результаты

Для сравнения эффективности алгоритма метаэвристического программирования для эволюционного синтеза при использовании десяти различных биоинспирированных алгоритмов с другими алгоритмами генетического программирования для задачи поиска аналитического описания модели (символьной регрессии) на основе заданных экспериментальных данных, множества переменных, базовых функций и операций были выбраны одиннадцать тестовых функций:

$$\begin{aligned}
&\text{Test1} : x^4 + x^3 + x^2 + x, \\
&\text{Test2} : \sin(x^2 + x^4), \\
&\text{Test3} : \sin(\exp(\sin(\exp(\sin(x))))), \\
&\text{Test4} : \sin(x^3) + e^x, \\
&\text{Test5} : \sin(2x) + 1/x - x^3, \\
&\text{Test6} : (x + a)/\sin(2x - 4), \\
&\text{Test7} : 2\sin(x)\exp(a), \\
&\text{Test8} : \sin(x) + \sin(a^2), \\
&\text{Test9} : x^5 - 2x^3 + x, \\
&\text{Test10} : \sin(x) + \sin(x^2 + x), \\
&\text{Test11} : \sin(x^2)\exp(x) - 1.
\end{aligned}$$

В экспериментах использовались значения каждой функции в 20-ти случайных точках в диапазоне (0,2), множество базовых функций $F_1 = \{+, *, \sin, \exp\}$ для тестовых функций Test1–Test4; базовых функций $F_2 = \{+, -, *, /, \sin, \exp\}$ для тестовых функций Test5–Test11; терминальные символы $T_1 = \{x\}$ для тестовых функций Test1–Test5 и Test10–Test11; терминальные символы $T_2 = \{x, a, 2, 5\}$ для функций Test6–Test9.

Использовались следующие параметры алгоритмов: размер популяции 300, вероятность кроссовера 0.80, вероятность мутации 0.15, максимальное число генераций 250. Длина хромосом равна 30 команд, для ГП задается начальная глубина дерева выражения, равная 6, для ДГП задаются размеры функциональной сети $32 = 2 \cdot 16$ и максимальное число генераций эволюционной стратегии (1+4)ES для функций Test1–Test5 — 10000, для функции Test6–Test12 — 75000. Для каждого алгоритма и каждой тестовой функции программа исполнялась 60 раз и результаты усреднялись. Отметим, что в описанных экспериментах были выбраны значения параметров используемых алгоритмов, рекомендуемые их авторами в приведенных статьях и библиотеках программ и позволяющие провести корректное сравнение их эффективности. Отдельно оптимизация подбора параметров алгоритмов не производилась.

В экспериментах с алгоритмом метаэвристического программирования для эволюционного синтеза использовались следующие десять биоинспирированных алгоритмов оптимизации: генетический алгоритм — две модификации — для векторов либо действительных чисел (mpGA_r), либо целых чисел (mpGA_i) [17], дифференциальной эволюции (mpDE) [18, 19], алгоритм оптимизации роем частиц (mpPSO) [20, 7], алгоритм колонии пчел (mpABC) [21, 7], алгоритм оптимизации на основе преподавания и обучения (mpTLB) [22] и его две модификации: (mpMTLB) [23] и (mpSPM) [24], эволюционная стратегия с адаптацией матрицы ковариаций (mpCMA) [25], алгоритм поиска на основе теплопередачи (mpSHT) [26].

Алгоритм метаэвристического программирования для эволюционного синтеза и другие тестируемые алгоритмы были реализованы на ресурсах Суперкомпьютерного центра Новосибирского государственного университета. Эксперименты выполнялись на процессорах Intel Xeon X5670, 2.93 GHz (Westmere). Программы написаны на языке системы Matlab. Для сравнения использовались следующие реализации рассматриваемых алгоритмов в системе Matlab:
ГП — GPLAB v.4.02 (`http://gplab.sourceforge.net/download.html`),
ГЭ — GEM-v0.2 (`http://ncra.ucd.ie/Software.html`),
ДГП — cgpmatlab (`http://www.cartesiangp.co.uk/resources/cgpmatlab.zip`).

Один из основных показателей, используемых для измерения эффективности эволюционных алгоритмов, — это вероятность (частота) успеха, вероятность того, что алгоритм обнаружил (синтезировал) выражение, в точности совпадающее с эталонной функцией, т. е. отношение числа успешных опытов, когда алгоритм обнаружил правильное выражение, к общему числу опытов с использованием заданных параметров. В табл. 2 представлены вероятности (частоты) успеха для различных алгоритмов, где $\overline{p}$ — средняя вероятность успеха для каждого алгоритма по множеству тестовых функций. По степени увеличения средней вероятности успеха алгоритмы можно упорядочить в следующей последовательности: ГЭ < ГП < ДГП < mpPSO < mpABC < mpCMA < mpSHT < mpSPM < mpTLB < mpMTLB < mpDE < mpGA_i < mpGA_r. Из таблицы видно, что вероятность успеха у МПЭС во многих случаях более чем в два раза выше, чем у других алгоритмов (и во всех остальных случаях вероятность успеха у МПЭС не меньше). Это объясняется способностью МПЭС представлять несколько выражений в одной хромосоме, что приводит к более высокому шансу обнаружить решение.

Таблица 2. Частота успешного поиска тестовых функций

Алгоритм	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Test11	$\bar{p}$
ГП	0.32	0.05	0.9	0.67	0	0	0.03	0.05	0.08	0	0	0.19
ГЭ	0.03	0.05	0.88	0.03	0	0	0.17	0.21	0	013	0	0.136
ДГП	0.27	0.35	0.25	0.12	0	0	0.71	0.88	0.17	0.5	0.25	0.317
mpGA$_i$	1	1	1	1	0.35	0.03	1	1	1	1	0.37	0.790
mpGA$_r$	1	1	1	1	035	0.04	1	1	1	1	0.32	0.792
mpDE	1	1	1	1	0.03	0.05	1	1	1	1	0.46	0.776
mpPSO	0.99	1	0.85	0.99	0.03	0.01	0.26	0.91	0.94	0.6	0.06	0.60
mpTLB	1	1	1	1	0.03	0.01	1	1	1	1	0.33	0.76
mpCMA	1	1	1	1	0	0.02	0.89	0.99	0.86	0.57	0.09	0.67
mpSHT	1	1	1	1	0.01	0.01	0.96	1	0.92	0.85	0.17	0.72
mpSPM	1	1	1	1	0.01	0.01	0.97	1	0.96	0.97	0.05	0.72
mpMTLB	1	1	1	1	0.04	0	1	1	1	1	0.28	0.76
mpABC	0.99	1	1	0.99	0,07	0.04	0.92	0.94	0.37	0.85	0.1	0.66

Второй показатель, используемый для оценки эффективности эволюционных алгоритмов в данной работе, — это среднее время поиска тестовых функций, т. е. среднее время исполнения алгоритма до того момента, как алгоритм обнаружил (синтезировал) заданное выражение либо выполнил требуемое число поколений.

В табл. 3 представлено среднее время поиска тестовых функций для различных алгоритмов, где $\bar{t}$ — среднее время поиска для каждого алгоритма по множеству тестовых функций. По степени уменьшения времени решения алгоритмы можно упорядочить в следующей последовательности: ГП > ГЭ > ДГП > mpCMA > mpABC > mpPSO > mpTLB > mpSPM > mpMTLB > mpDE > mpSHT > mpGA$_i$ > mpGA$_r$. Из данной таблицы видно, что МПЭС имеет меньшее время поиска решения (более чем на порядок в большинстве случаев).

Таблица 3. Среднее время поиска тестовых функций

Алгоритм	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Test11	$\bar{t}$
ГП	783.2	309.7	23.6	118.6	33.2	11265	4425	4746	5540	10468	9377	4280.8
ГЭ	42.4	43.8	19.6	43.8	42.7	225.2	220.8	224.5	193.6	211	312	143.6
ДГП	3.9	3.8	4.4	4.1	5.1	277	135.7	106.7	217.9	203.3	214.7	106.96
mpGA$_i$	0.39	0.25	0.31	0.35	1.4	3,77	0.348	0.336	0.91	0.35	2.95	3.38
mpGA$_r$	0.27	0.26	0.26	0.3	1.3	3.6	0.32	0.33	0.88	0.49	2.5	3.32
mpDE	0.48	0.35	0.35	0.62	21.7	24.2	1.59	1.65	6.96	3.16	21.5	7.5
mpPSO	0.72	0.10	7.34	1.17	36.8	40.0	31.1	5.0	3.99	20.9	39.2	16.9
mpTLB	0.35	0.28	0.45	0.69	37.3	38.7	3.08	2.3	5.12	5.1	33.4	11.5
mpCMA	1.65	0.68	0.99	1.4	81.8	76.7	12.8	6.9	28.6	45.3	78.2	30.5
mpSHT	0.21	0.18	0.13	0.9	15.4	16.9	3.62	2.55	4.75	6.5	15.3	6.04
mpSPM	0.38	0.33	0.42	1.3	21.3	21.4	5.84	3.74	7.6	7.8	21.0	8.28
mpMTLB	0.31	0.2	0.4	0.66	23.0	27.4	3.2	1.94	2.95	4.8	23.7	8.05
mpABC	7.4	3.18	5.5	7.2	34.5	35.2	14.3	11.2	34.2	18.5	37.9	19.0

Это можно объяснить: во-первых, простыми генетическими операциями и структурами у МПЭС; во-вторых, прямым исполнением инструкций во время декодирования хромосом "на лету" без получения всего выражения отдельно в виде текстовой строки и ее интерпретации, как в других алгоритмах; и в-третьих, использованием во время вычислений инструкций векторных операций для всего набора входных данных.

5. Заключение

Представлен метод метаэвристического программирования для эволюционного синтеза моделей, объединяющий преимущества биоинспирированных алгоритмов оптимизации и генетического программирования. Предложенный метод осуществляет построение нелинейных моделей (математических выражений, функций, алгоритмов, программ) на основе заданных экспериментальных данных, множества переменных, базовых функций и операций. Разработанный алгоритм использует в совокупности линейное представление хромосомы в виде вектора действительных чисел, простые операции при декодировании генотипа в фенотип для интерпретации хромосомы как последовательности команд, многовариантный метод для представления множества моделей (выражений) с помощью одной хромосомы, векторизацию вычислений при определении целевой (фитнес) функции. Предложенный алгоритм метаэвристического программирования был реализован на основе десяти различных биоинспирированных алгоритмов оптимизации и было проведено его сравнение со стандартным алгоритмом генетического программирования, алгоритмом грамматической эволюции и алгоритмом декартового генетического программирования. Проведенные эксперименты показали существенное преимущество предложенного подхода по сравнению с указанными алгоритмами как по времени поиска решения (более чем на порядок в большинстве случаев), так и по вероятности нахождения заданной функции (модели) (во многих случаях более чем в два раза).

Литература

1. **Kitano H.** Artificial intelligence to win the Nobel prize and beyond: creating the engine for scientific discovery // AI Magazine. — 2016. — Vol. 37, № 1. — P. 39–49.
2. **Koza J.R.** Genetic Programming II: Automatic Discovery of Reuseable Programs. — MIT Press, 1996.
3. **Koza J.R.** Genetic programming as a means for programming computers by natural selection // Statistics and Computing. — 1994. — Vol. 4. — P. 87–112.
4. **Langdon W.B., Poli R.** Foundations of Genetic Programming. — Springer-Verlag, 2002.
5. **Poli R., Langdon W.B., McPhee N.F.** A Field Guide to Genetic Programming. — San Francisco, California, USA: Lulu.com, 2008.
6. **Емельянов В.В., Курейчик В.В., Курейчик В.М.** Теория и практика эволюционного моделирования.— М.: ФИЗМАТЛИТ, 2003.
7. **Карпенко А.П.** Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2014.
8. **Монахов О.Г., Монахова Э.А.** О параллельном алгоритме многовариантного эволюционного синтеза нелинейных моделей // Тр. 12-й Международной Азиатской школы-семинара "Проблемы оптимизации сложных систем". — 2016. — С. 390–395.
9. **O'Neill M., Ryan C.** Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. — Kluwer Academic Publishers, 2003.

10. **Ryan C., Collins J., O'Neill M.O.** Grammatical evolution: Evolving programs for an arbitrary language / W. Banzhaf, R. Poli, M. Schoenauer, T.C. Fogarty // Genetic Programming. EuroGP. — Berlin, Heidelberg: Springer, 1998. — Vol. 1391. — P. 83–96. — (Lecture Notes in Computer Science.)
11. **Miller J.F.** Cartesian Genetic Programming. — Springer, 2011.
12. **Miller J.F., Thomson P.** Cartesian Genetic Programming // Proc. of the 3rd European Conference on Genetic Programming. — Springer, 2000. — P. 121–132. — (Lecture Notes in Computer Science 1802.)
13. **Дивеев А.И.** Метод сетевого оператора. — М.: ВЦ РАН, 2010.
14. **Монахова Э.А., Монахов О.Г.** Поиск рекордных циркулянтных графов с использованием параллельного генетического алгоритма // Дискретный анализ и исследование операций. — 2015. — Т. 22, № 6. — С. 29–39.
15. **Oltean M.** Multi Expression Programming. — Romania: Babes-Bolyai Univ., 2006. — (Technical Report.)
16. **Монахов О.Г.** Метод многовариантного эволюционного синтеза моделей на основе темплейтов // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. — 2013. — № 3. — С. 269–282.
17. **Монахов О.Г., Монахова Э.А.** Параллельный алгоритм многовариантного эволюционного синтеза нелинейных моделей // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2017. — Т. 20, № 2. — С. 169–180. Перевод: Monakhov O.G., Monakhova E.A. A parallel algorithm of multi-variant evolutionary synthesis of nonlinear models // Numerical Analysis and Applications. — 2017. — Vol. 10,№ 2. — P. 140–148.
18. **Storn R., Price K.** Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces // J. Global Optimization. — 1997. — Vol. 11, № 4. — P. 341–359.
19. **Zaheer Hira, Pant Millie, Kumar Sushil, Monakhov Oleg, Monakhova Emilia, Deep Kusum** A new guiding force strategy for differential evolution // Int. J. of System Assurance Engineering and Management. — 2017. — Vol. 8, suppl. 4. — P. 2170–2183.
20. **Bonyadi M.R., Michalewicz Z.** Particle swarm optimization for single objective continuous space problems: a review // Evolutionary Computation. — 2017. — Vol. 25, № 1. — P. 1–54.
21. **Karaboga D., Akay B.** A comparative study of artificial bee colony algorithm // Applied Mathematics and Computation. — 2009. — Vol. 214. — P. 108–132.
22. **Rao R.V., Savsani V.J., Vakharia D.P.** Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems // Computer-Aided Design. — 2011. — Vol. 43, № 2. — P. 303–315.
23. **Crepinsek M., Liu S-H., Mernik L.** A note on teaching–learning-based optimization algorithm // Information Sciences. — 2012. — Vol. 212. — P. 79–93.
24. **Kommadath R., Sivadurgaprasad C., Kotecha P.** Single phase multi-group teaching learning algorithm for single objective real-parameter numerical optimization (CEC2016) // IEEE Congress on Evolutionary Computation (CEC). — 2016. — P. 1165–1172.
25. **Hansen N, Ostermeier A.** Completely derandomized self-adaptation in evolution strategies // Evolutionary Computation. — 2001. — Vol. 9, № 2. — P. 159–195.
26. **Maharana D., Kotecha P.** Simultaneous heat transfer search for computationally expensive numerical optimization // IEEE Congress on Evolutionary Computation (CEC). — 2016. — P. 2982–2988.

Поступила в редакцию 4 декабря 2018 г.
После исправления 5 апреля 2019 г.
Принята к печати 16 июля 2020 г.

Литература в транслитерации

1. **Kitano H.** Artificial intelligence to win the Nobel prize and beyond: creating the engine for scientific discovery // AI Magazine. — 2016. — Vol. 37, № 1. — P. 39–49.
2. **Koza J.R.** Genetic Programming II: Automatic Discovery of Reuseable Programs. — MIT Press, 1996.
3. **Koza J.R.** Genetic programming as a means for programming computers by natural selection // Statistics and Computing. — 1994. — Vol. 4. — P. 87–112.
4. **Langdon W.B., Poli R.** Foundations of Genetic Programming. — Springer-Verlag, 2002.
5. **Poli R., Langdon W.B., McPhee N.F.** A Field Guide to Genetic Programming. — San Francisco, California, USA: Lulu.com, 2008.
6. **Emel'yanov V.V., Kureichik V.V., Kureichik V.M.** Teoriya i praktika evolyutsionnogo modelirovaniya.— M.: FIZMATLIT, 2003.
7. **Karpenko A.P.** Sovremennye algoritmy poiskovoi optimizatsii. Algoritmy, vdokhnovlennye prirodoi. — M.: Izd-vo MGTU im. N.E. Baumana, 2014.
8. **Monakhov O.G., Monakhova E.A.** O parallel'nom algoritme mnogovariantnogo evolyutsionnogo sinteza nelineinykh modelei // Tr. 12-i Mezhdunarodnoi Aziatskoi shkoly-seminara "Problemy optimizatsii slozhnykh sistem". — 2016. — S. 390–395.
9. **O'Neill M., Ryan C.** Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. — Kluwer Academic Publishers, 2003.
10. **Ryan C., Collins J., O'Neill M.O.** Grammatical evolution: Evolving programs for an arbitrary language / W. Banzhaf, R. Poli, M. Schoenauer, T.C. Fogarty // Genetic Programming. EuroGP. — Berlin, Heidelberg: Springer, 1998. — Vol. 1391. — P. 83–96. — (Lecture Notes in Computer Science.)
11. **Miller J.F.** Cartesian Genetic Programming. — Springer, 2011.
12. **Miller J.F., Thomson P.** Cartesian Genetic Programming // Proc. of the 3rd European Conference on Genetic Programming. — Springer, 2000. — P. 121–132. — (Lecture Notes in Computer Science 1802.)
13. **Diveev A.I.** Metod setevogo operatora. — M.: VTS RAN, 2010.
14. **Monakhova E.A., Monakhov O.G.** Poisk rekordnykh tsirkulyantnykh grafov s ispol'zovaniem parallel'nogo geneticheskogo algoritma // Diskretnyi analiz i issledovanie operatsii. — 2015. — T. 22, № 6. — S. 29–39.
15. **Oltean M.** Multi Expression Programming. — Romania: Babes-Bolyai Univ., 2006. — (Technical Report.)
16. **Monakhov O.G.** Metod mnogovariantnogo evolyutsionnogo sinteza modelei na osnove templeitov // Nauka i obrazovanie: nauchnoe izdanie MGTU im. N.E. Baumana. — 2013. — № 3. — C. 269–282.
17. **Monakhov O.G., Monakhova E.A.** Parallel'nyi algoritm mnogovariantnogo evolyutsionnogo sinteza nelineinykh modelei // Sib. zhurn. vychisl. matematiki / RAN. Sib. otd-nie. — Novosibirsk, 2017. — T. 20, № 2. — S. 169–180. Perevod: Monakhov O.G., Monakhova E.A. A parallel algorithm of multi-variant evolutionary synthesis of nonlinear models // Numerical Analysis and Applications. — 2017. — Vol. 10,№ 2. — P. 140–148.
18. **Storn R., Price K.** Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces // J. Global Optimization. — 1997. — Vol. 11, № 4. — P. 341–359.
19. **Zaheer Hira, Pant Millie, Kumar Sushil, Monakhov Oleg, Monakhova Emilia, Deep Kusum** A new guiding force strategy for differential evolution // Int. J. of System Assurance Engineering and Management. — 2017. — Vol. 8, suppl. 4. — P. 2170–2183.

20. **Bonyadi M.R., Michalewicz Z.** Particle swarm optimization for single objective continuous space problems: a review // Evolutionary Computation. — 2017. — Vol. 25, № 1. — P. 1–54.

21. **Karaboga D., Akay B.** A comparative study of artificial bee colony algorithm // Applied Mathematics and Computation. — 2009. — Vol. 214. — P. 108–132.

22. **Rao R.V., Savsani V.J., Vakharia D.P.** Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems // Computer-Aided Design. — 2011. — Vol. 43, № 2. — P. 303–315.

23. **Crepinsek M., Liu S-H., Mernik L.** A note on teaching–learning-based optimization algorithm // Information Sciences. — 2012. — Vol. 212. — P. 79–93.

24. **Kommadath R., Sivadurgaprasad C., Kotecha P.** Single phase multi-group teaching learning algorithm for single objective real-parameter numerical optimization (CEC2016) // IEEE Congress on Evolutionary Computation (CEC). — 2016. — P. 1165–1172.

25. **Hansen N, Ostermeier A.** Completely derandomized self-adaptation in evolution strategies // Evolutionary Computation. — 2001. — Vol. 9, № 2. — P. 159–195.

26. **Maharana D., Kotecha P.** Simultaneous heat transfer search for computationally expensive numerical optimization // IEEE Congress on Evolutionary Computation (CEC). — 2016. — P. 2982–2988.