COMPARATIVE EVALUATION OF ALGORITHMS FOR AUTOMATIC CONSTRUCTION OF NONLINEAR MODELS BASED ON METAHEURISTIC PROGRAMMING WITH GENE EXPRESSION

СРАВНИТЕЛЬНАЯ ОЦЕНКА АЛГОРИТМОВ АВТОМАТИЧЕСКОГО ПОСТРОЕНИЯ НЕЛИНЕЙНЫХ МОДЕЛЕЙ НА ОСНОВЕ МЕТАЭВРИСТИЧЕСКОГО ПРОГРАММИРОВАНИЯ С ЭКСПРЕССИЕЙ ГЕНОВ.

Oleg Monakhov (О.Г. Монахов)

Emilia Monakhova (Э.А. Монахова)

ICMMG SB RAS, (ИВМиМГ СО РАН)

Novosibirsk (Новосибирск), Russia

monakhov@rav.sscc.ru

The work has been carried out under budget project of ICMMG SB RAS N 0251-2021-0005.

План

1. Введение, постановка задачи

2. Представление решений в системах генетического программирования (ГП)
   - Этапы моделирования процесса эволюции
   - Представление данных
   - Операторы алгоритма ГП

3. Алгоритм метаэвристического программирования с экспрессией генов для эволюционного синтеза
   - Представление данных
   - Операторы алгоритма
   - Итерационный процесс

4. Экспериментальные результаты и сравнение алгоритмов

5. Заключение

Введение, постановка задачи

Рассматривается проблема построения нелинейных моделей, представленных в виде математических выражений, функций, формул, алгоритмов, программ, на основе заданных экспериментальных данных, множества 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, то есть требуется подобрать такую функцию ***Y=f(X)***, которая отображает зависимость ***Y*** от ***X*** с минимальной погрешностью (иногда эту задачу называют символьной регрессией, идентификацией системы, equation discovery). Поиск осуществляется на основе заданного множества базовых функций, операций и переменных, с помощью которых автоматически создаются аналитические выражения (формулы), представляющие модель, и компьютерные программы для их вычисления.

Представлен новый подход *метаэвристического программирования* (МП) с экспрессией генов для синтеза нелинейных моделей, имеющий во многих случаях более высокую эффективность эволюционного поиска по сравнению с известными ранее системами генетического программирования.

Представление решений в системе генетического программирования

Этапы моделирования процесса эволюции в системах генетического программирования заключаются в следующем:

1. **Создание первоначальной популяции из случайно сгенерированных решений (хромосом). Отметим, что решение в хромосоме представлено как правило в закодированном виде - в виде генотипа.**

2. **Оценка популяции по фитнес-функции (fitness function, функция пригодности, функция приспособленности), 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 следующих эволюционных операторов:**
   - **3.1. Выбор лучшего решения в популяции и копирование его в следующее поколение.**
   - **3.2. Создание новых хромосом методом кроссовера.**
   - **3.3. Создание новых хромосом методом мутации.**

4. **Повторение п.2 и п.3, пока решение по заданному критерию не будет найдено или не будет достигнуто максимальное число поколений.**

Genetic Programming (GP)

Поиск решения в процессе эволюции осуществляется на основе заданного множества базовых (элементарных) функций (function set, например, $F_1 = \{+, -, /, *, sin, exp\}$), и заданного множества свободных, проблемно-ориентированных переменных и констант (терминальное множество - terminal set, например, $T_1 = \{x, y, a1, 2, 3.14\}$) из которых строится требуемое математическое выражение (модель, программа).

При стандартном генетическом программировании (ГП) решение представляется в виде дерева. 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 на другую, случайно выбранную величину из множества допустимых значений.

Проблемы GP:

1. Эффект неоправданного роста выражений (bloat) и появления функционально лишних частей (интронов), что ведет к снижению эффективности эволюционного поиска.

2. Появление некорректных потомков после случайных операторов.

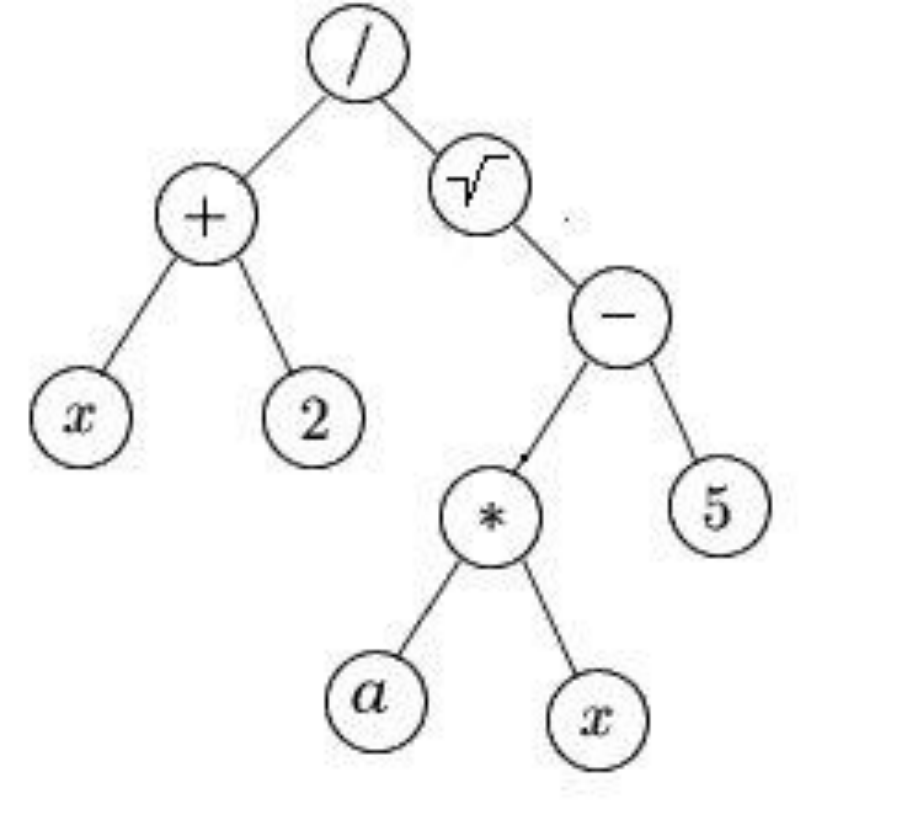
Genetic Programming (GP)

Tree reresentation for GP: function (formula),

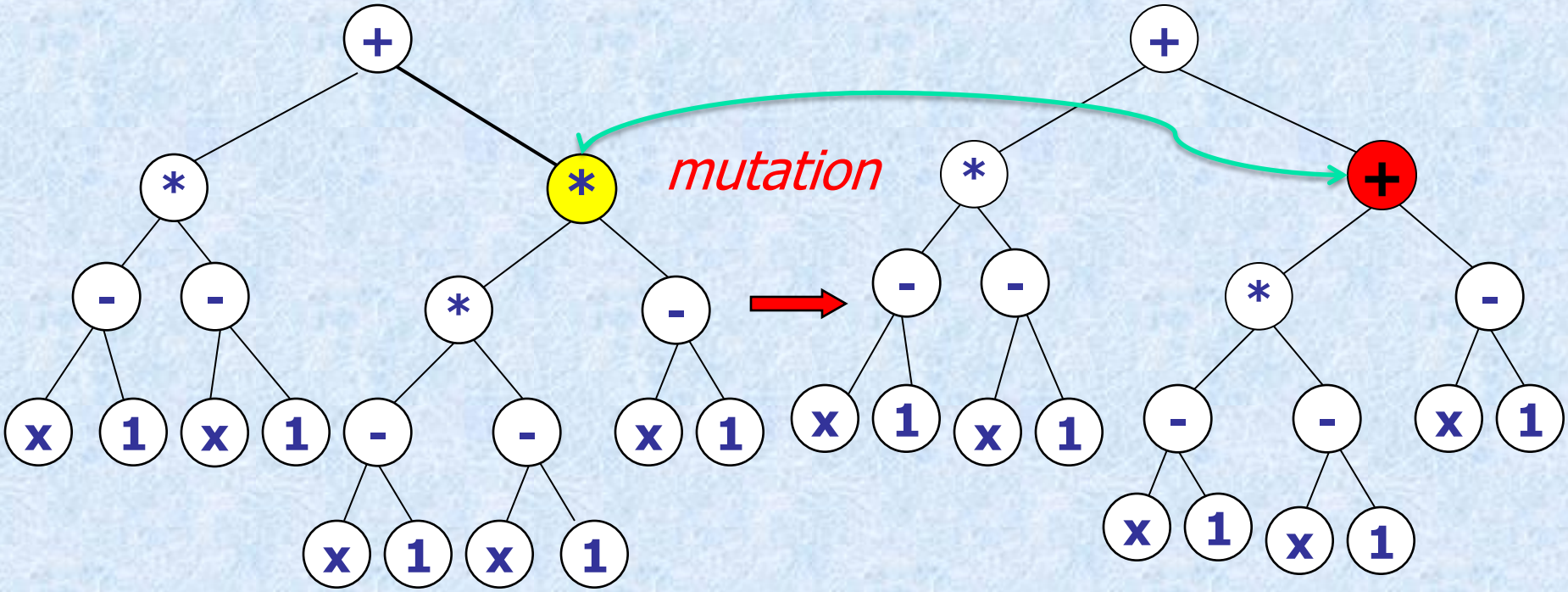
$$f_n = (x+2)/\sqrt{ax-5}$$

with terminals (variables/constants): $T = \{x, a, 2{,}5\}$

and nonterminals (operations/primitive functions): $F = \{+, -, /, *, \sqrt{\ }\}$



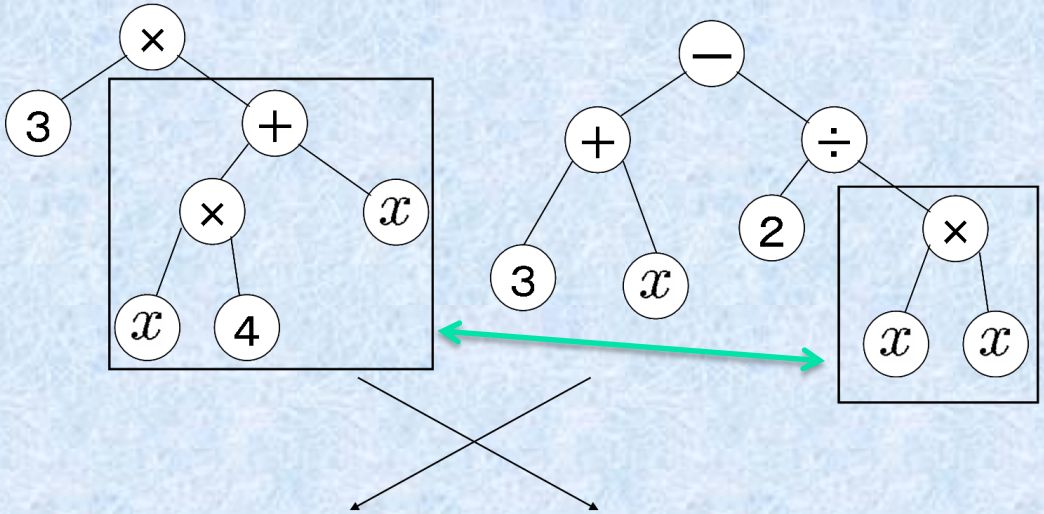
Genetic Programming (Point Mutation)



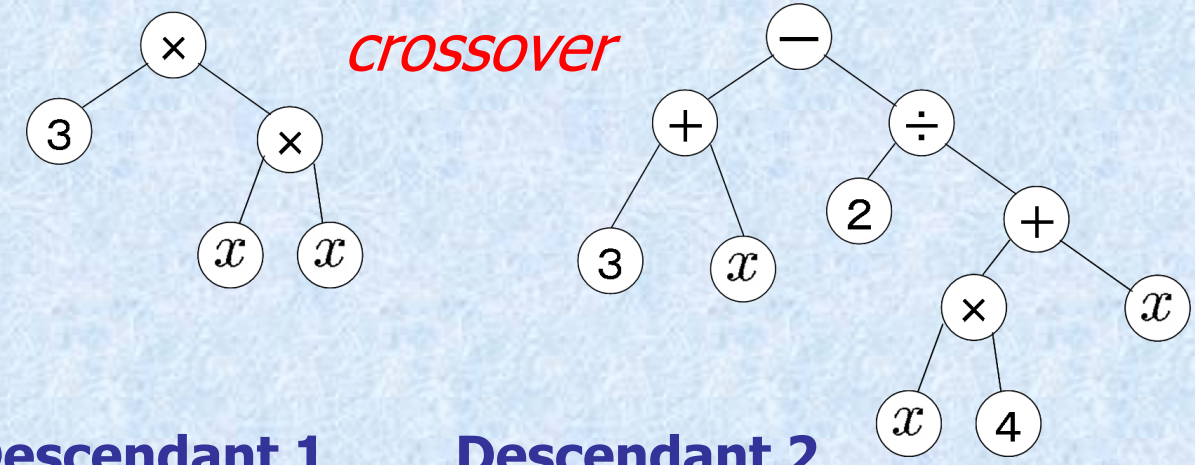
Genetic Programming (One-point Crossover)

Subtree Exchange Crossover

Parent 1 **Parent 2**



crossover



Descendant 1 **Descendant 2**

Gene Expression Programming (GEP)

Function (formula) reresentation for GEP:

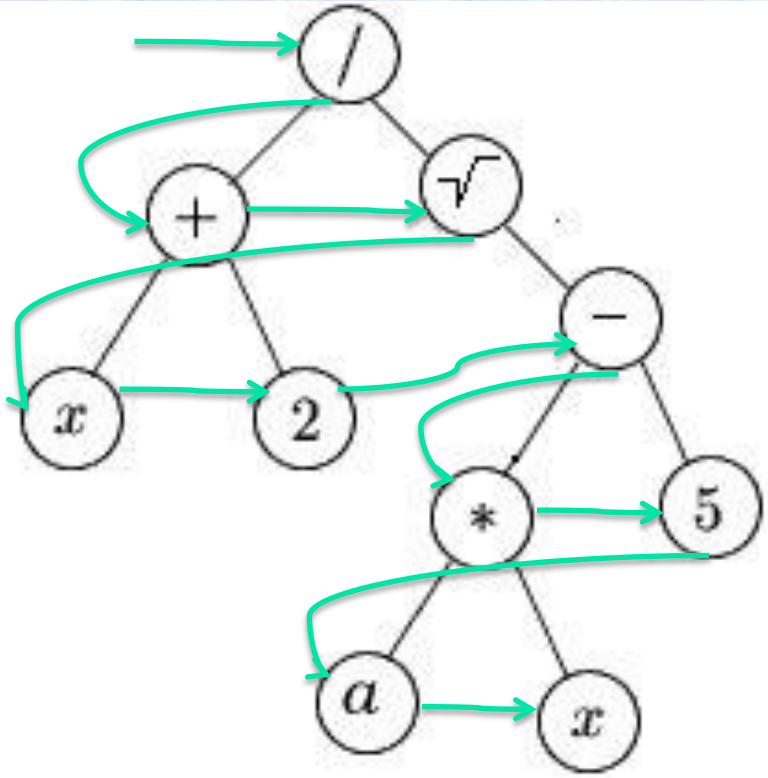
$f_n = (x+2)/\sqrt{ax-5} \rightarrow$ {/+Qx2-*5 ax3axx4aa}

tail = head * (n-1) +1

| head | tail |

with terminals (variables/constants): $T = \{x, a, 2, 5\}$

and nonterminals (operations/primitive functions): $F = \{+, -, /, *, \sqrt{\ }\}$



Обход дерева в ширину

1. {/+Qx2-*5 ax 3axx4aa}

2. {/+Qx2-*5 ax 3axx4aa}

3. {/+Qx2-*5 ax 3axx4aa}

4. {/+Qx2-*5 ax 3axx4aa}

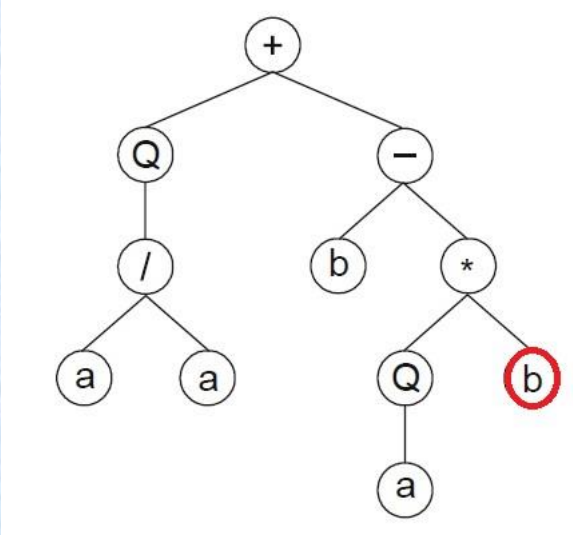
5. {/+Qx2-*5 ax 3axx4aa}

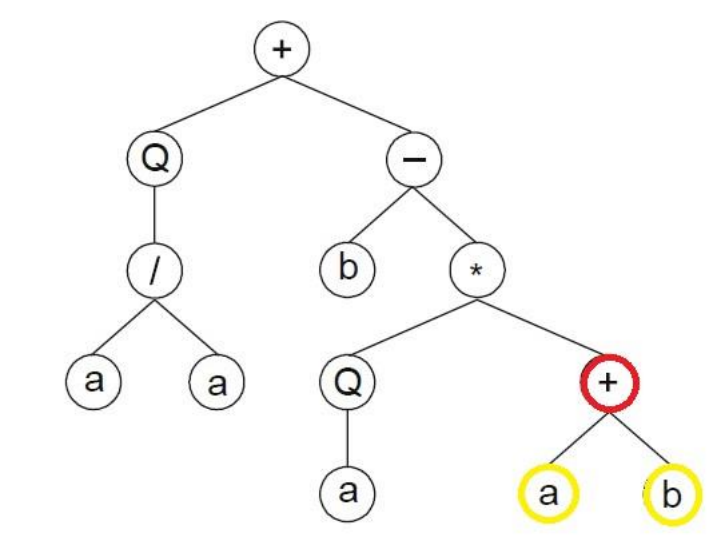
[1] C. Ferreira, Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, (2nd ed.), Springer-Verlag, Heidelberg, 2006

Mutation and crossover for GEP

Mutation:

{+Q-/b*aaQb aabaabbaaab} => {+Q-/b*aaQ+ aabaabbaaab}





One-point crossover:

{-b+Qbbabb/aQbbbaab} => {-b+/ababb-ba-abaaa}

{/-a/ababb-ba-abaaa} => {/-aQbbabb/aQbbbaab}

Two-point crossover:

{-b+Qbbabb/aQbbbaab} => {-b+/ababb-bQbbbaab}

{/-a/ababb-ba-abaaa} => {/-aQbbabb/aa-abaaa}

Алгоритм метаэвристического программирования с экспрессией генов для синтеза нелинейных моделей.

Алгоритм *метаэвристического программирования для синтеза нелинейных моделей* (МП) основан на эволюционных вычислениях (биоинспирированных, метаэвристических алгоритмах) и моделировании процесса естественного отбора в популяции особей, каждая из которых представляет точки в пространстве решений задачи оптимизации, а не единственное решение как в ГП. Особи (агенты, муравьи, пчелы, частицы, …) являются структурами данных - хромосомами, последовательностями (векторами) действительных чисел, которые представляют в закодированном виде математические выражения (формулы, программы). Каждая популяция является множеством хромосом и каждая хромосома, в данном алгоритме, определяет множество выражений (формул), образующихся из нее после декодирования.

Основная идея алгоритма синтеза состоит в эволюционном преобразовании множества хромосом (формул) в процессе естественного отбора с целью выживания "сильнейшего". В нашем случае этими особями являются выражения, имеющие наименьшее значение целевой функции. Алгоритм начинается с генерации начальной популяции. Все особи в этой популяции создаются случайно, затем отбираются наилучшие особи путем декодирования генотипа (хромосомы) в фенотип (выражение) и вычисления фитнес-функции. Для создания популяции следующего поколения (следующей итерации), новые особи формируются с помощью операций миграции особей используемого биоинспирированного алгоритма, в ГА это операции селекции (отбора), мутации и кроссовера.

0.11	0.14	0.97	0.13	0.55	…	0.91

Блок-схема алгоритма метаэвристического программирования с экспрессией генов для синтеза нелинейных моделей (МП)

Множество базовых функций $F = \{+, -, *, \ldots, \sqrt{\ }\}$ **и терминальное множество** $T = \{x, a, \ldots, 5\}$

Правила декодирования хромосомы в функцию:

$(\{m_{1j}\}, \{m_{2j}\}, \{m_{3j}\}) =>$

$h_j = \lfloor m_{1j} * (|F| + |T|) \rfloor$; *head*

$t_j = \lfloor (m_{2j} * |T|) \rfloor$; *tail*

$c_j = \lfloor (m_{3j} * |C|) \rfloor$, *const*

Начальная популяция P

(случайные вектора в (0,1))

<0.81, 0.39,..., 0.37>,

<0.95, 0.047,..., 0.10>,

...

<0.20, 0.90,..., 0.063>

Отображение хромосом P_i **в функции:**

$\{/+Qx2 -* 5ax3axx4aa\} \rightarrow (x+2)/e^{ax-5}$

Новая популяция P′:

< 0.26, 0.98,..., 0.25 >,

< 0.66, 0.75,..., 0.039 >,

...

< 0.78, 0.27, ..., 0.99>

Экспериментальные данные

(X′,Y′)

Вычисление фитнес-функции хромосом:

$FF = \sum_{i=1}^{N} (f(X_i) - Y_i)^2,$

Метаэвристический (биоинспирированный) алгоритм:

< мутация> ,

< кроссовер> ,

<селекция>,

<миграция>

ВЫХОД

Целевая функция

Целевая функция - фитнес-функция (функция качества, функция пригодности, функция приспособленности) FF **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**

$$X_i,\ 1 \le i \le N: \quad FF = \sum_{i=1}^{N} (f(X_i) - Y_i)^2,$$

где N **- число экспериментальных данных. Целью алгоритма является поиск минимума** FF **. На практике, если получалось несколько решений с одинаковым значением целевой функции, то выбиралось решение с меньшей оценкой структурной сложности, то есть с меньшей общей длиной (суммой числа элементов) формул решения.**

Алгоритм декодирования хромосомы (декодирование генотипа в фенотип)

- В предлагаемом методе метаэвристического программирования с экспрессией генов предлагается единый подход к декодированию основных структур данных, хромосом, для различных природно-инспирированных алгоритмов. Последовательность действительных чисел (генотип) делится на три группы элементов (для головы, хвоста и констант): $(\{m_{1j}\},\{m_{2j}\},\{m_{3j}\})$, $0 < m_{ij} < 1$; $|\{m_{1j}\}| = h$; $|\{m_{2j}\}| = t$; $|\{m_{3j}\}| = t$. Каждая такая группа интерпретируется разными алфавитами, которые рассчитываются по следующим формулам:

$(\{m_{1j}\},\{m_{2j}\},\{m_{3j}\}) =>$

$h_j = \lfloor m_{1j} * (|F| + |T|) \rfloor$; *head*

$t_j = \lfloor (m_{2j} * |T|) \rfloor$; *tail*

$c_j = \lfloor (m_{3j} * |C|) \rfloor$, *const*

где */F/* - число базовых функций,

/T/ - число терминалов.

/C/ - мощность множества констант.

Многовариантное декодирование:

*{+Qx2-*5 ax3axx4aa}* $=> x + \sqrt{2}$

*{Qx2-*5 ax3axx4aa}* $=> \sqrt{x}$

*{x2-*5 ax3axx4aa}* $=> x$

*{2-*5 ax3axx4aa}* $=> 2$

*{-*5 ax3axx4aa}* $=> ax - 5$

*{*5 ax3axx4aa}* $=> 5a$

{5 ax3axx4aa} $=> 5$

$(0{,}32; 0{,}11; 0{,}53; 0{,}7; 0{,}94; \ldots, 0{,}8) \rightarrow$ *{/+Qx2-*5 ax3axx4aa}* $\rightarrow (x+2)/\sqrt{ax-5}$

Декодирование хромосомы приводит к тому, что функция представляется как интерпретируемое выражение. Но мы использовали многовариантный алгоритм для декодирования выражения в несколько подвыражений. В этом многовариантном алгоритме последовательно повторяем чтение символов головы начиная с каждого символа до конца головы. Это позволяет, в отличие от стандартного GEP, одновременно оценивать множество выражений

Метаэвристические (биоинспирированные) алгоритмы

Метаэвристические популяционные алгоритмы для поиска оптимальной модели - Metaheuristic Algorithms for Optimization - предназначены для глобальной оптимизации недифференцируемых, нелинейных, мультимодальных функций от многих переменных и использующие некоторые аналогии природоподобных процессов.

Используемые метаэвристические алгоритмы:

(1) Генетический алгоритм - Genetic Algorithm
(2) Дифференциальная эволюция - Differential Evolution
(3) Алгоритм оптимизации роем частиц - Particle Swarm Optimization
(4) Artificial Bee Colony (ABC),
(5) Continuous Ant Colony Optimization (ACOR),
(6) Bees Algorithm (BA),
(7) Biogeography-based Optimization (BBO),
(8) Covariance Matrix Adaptation Evolution Strategy (CMA-ES),
(9) Firefly Algorithm (FA),
(10) Harmony Search (HS),
(11) Imperialist Competitive Algorithm (ICA),
(12) Invasive Weed Optimization (IWO),
(13) Simulated Annealing (SA),
(14) Teaching-Learning-based Optimization (TLB)

Операторы алгоритма ГА (Кроссовер, селекция, новый элемент)

Оператор кроссовера (скрещивания) применяется к двум особям (родителям), случайно выбранным из текущей популяции с вероятностью $p_{cr} \in [0,1]$. Кроссовер состоит в порождении двух новых особей путем обмена частями хромосом родителей.

- **Оператор создания нового элемента (особи) состоит в генерации случайных значений параметров, описывающих особи. Это позволяет увеличить степень разнообразия особей при создании популяции.**

- **Оператор селекции (отбора). Вычисляются целевые функции новых векторов, полученных посредством мутации и кроссовера, при этом происходит *декодирование векторов (хромосом) в выражения* и программы описанным выше процессом. Если они имеют целевые функции меньше, чем некоторые векторы популяции, тогда 'наихудшие' векторы (с большим значением целевой функции) в популяции замещаются новыми 'наилучшими` (с меньшим значением целевой функции).**

Заметим, что только простейшие генетические операторы были использованы в алгоритме, но данный подход позволяет применять и более сложные генетические операторы.

Операторы алгоритма ГА (Оператор мутации) Mutation (GA)

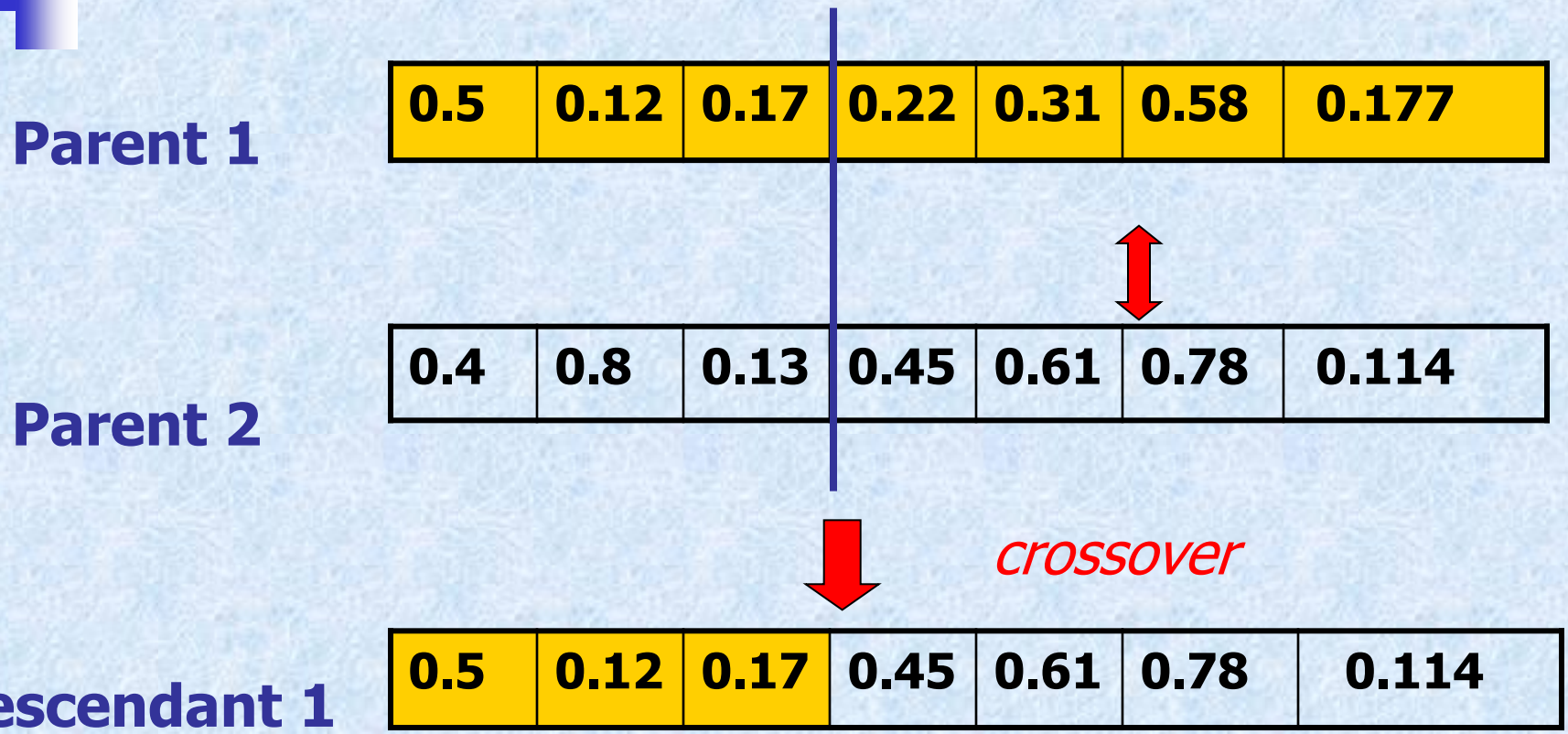
Оператор мутации применяется к особям из текущей популяции. Мутация хромосомы *Gen* состоит в изменении значения каждого ее параметра независимо от остальных с вероятностью $p_{mut} \in [0,1]$ на другую, случайно выбранную величину из множества допустимых значений.

0.5	**0.12**	**0.17**	**0.2**	**0.31**	**0.58**	**0.177**

mutation

0.4	**0.12**	**0.17**	**0.9**	**0.31**	**0.21**	**0.177**

Операторы алгоритма ГА (Кроссовер)
GA (One-point Crossover)



Differential Evolution (DE) for Metaheuristic Programming
Дифференциальная эволюция

Определим начальную популяцию, состоящую из случайных векторов вещественных чисел. Затем мы применяем генетические операторы мутации, кроссовера и отбора к этой популяции.

Инициализация. **На этом этапе M-мерных решений (векторов) генерируются случайным образом. Для всех решений производиться декодирование генотипа и вычисляются значения целевой функции.**

Мутация. **Принимая каждый вектор (один за другим) в качестве целевого вектора, выбираются три случайных вектора** $x_{r_1,G}$, $x_{r_2,G}$, $x_{r_3,G}$

С помощью целевых векторов и этих случайных векторов формируется новый вектор -мутантный вектор.

$$v_{i,G+1} = x_{r_1,G} + F\left(x_{r_2,G} - x_{r_3,G}\right),$$

вес F - коэффициентом масштабирования.

Кроссовер. **Вектор, сформированный после операции кроссовера с целевым вектором и мутантным вектором, называется пробный вектор.**

$$u_{j,i,G+1} = \left(u_{1,i,G+1}, u_{2,i,G+1} \ldots\ldots u_{D,i,G+1}\right),$$

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & if\ rand \le C_r \forall j = j_{rand} \\ x_{j,i,G} & otherwise\ . \end{cases}$$

C_r **- вероятность кроссовера.**

Селекция. **Декодирование генотипа и вычисление целевой функции. Проводиться сравнение между пробным вектором и целевым вектором, тот вектор, для которого целевая функция минимальна, будет выбираться для следующего поколения. Этот цикл мутации, кроссовера и отбора будет продолжаться до тех пор, пока не будет достигнут критерий остановки.** $x_{i,G+1}$

$$= \begin{cases} u_{i,G+1} & if\ f\left(u_{i,G+1}\right) \le f\left(x_{i,G}\right) \\ x_{i,G} & otherwise\ . \end{cases}$$

Particle Swarm Optimization (PSO) for Metaheuristic Programming

Алгоритм оптимизации роем частиц

Пусть f: $\mathbb{R}^n \to \mathbb{R}$ — целевая функция, которую требуется минимизировать, S — количество частиц в рое, каждой из которых сопоставлена координата $x_i \in \mathbb{R}^n$ в пространстве решений и скорость $v_i \in \mathbb{R}^n$. Пусть также p_i — лучшее из известных положений частицы i, а g — наилучшее известное состояние роя в целом. Тогда общий вид метода роя частиц таков.

Для каждой частицы $i = 1, \ldots, S$ сделать:

Сгенерировать начальное положение частицы с помощью случайного вектора $x_i \sim U(b_{lo}, b_{up})$, b_{lo} и b_{up} — нижняя и верхняя границы пространства решений соответственно.

Декодирование генотипа и вычисление целевой функции. Присвоить лучшему известному положению частицы его начальное значение: $p_i \leftarrow x_i$.

Если ($f(p_i) < f(g)$), то обновить наилучшее известное состояние роя: $g \leftarrow p_i$.

Присвоить значение скорости частицы: $v_i \sim U(-(b_{up}-b_{lo}), (b_{up}-b_{lo}))$.

Пока не выполнен критерий остановки (например, достижение заданного числа итераций или необходимого значения целевой функции), повторять:

Для каждой частицы $i = 1, \ldots, S$ сделать:

Сгенерировать случайные векторы $r_p, r_g \sim U(0,1)$.

Обновить скорость частицы: $v_i \leftarrow \omega v_i + \varphi_p r_p \times (p_i - x_i) + \varphi_g r_g \times (g - x_i)$, где операция $\times$ означает покомпонентное умножение.

Обновить положение частицы переносом x_i на вектор скорости: $x_i \leftarrow x_i + v_i$. Заметим, что этот шаг выполняется вне зависимости от улучшения значения целевой функции. **Декодирование генотипа и вычисление целевой функции.**

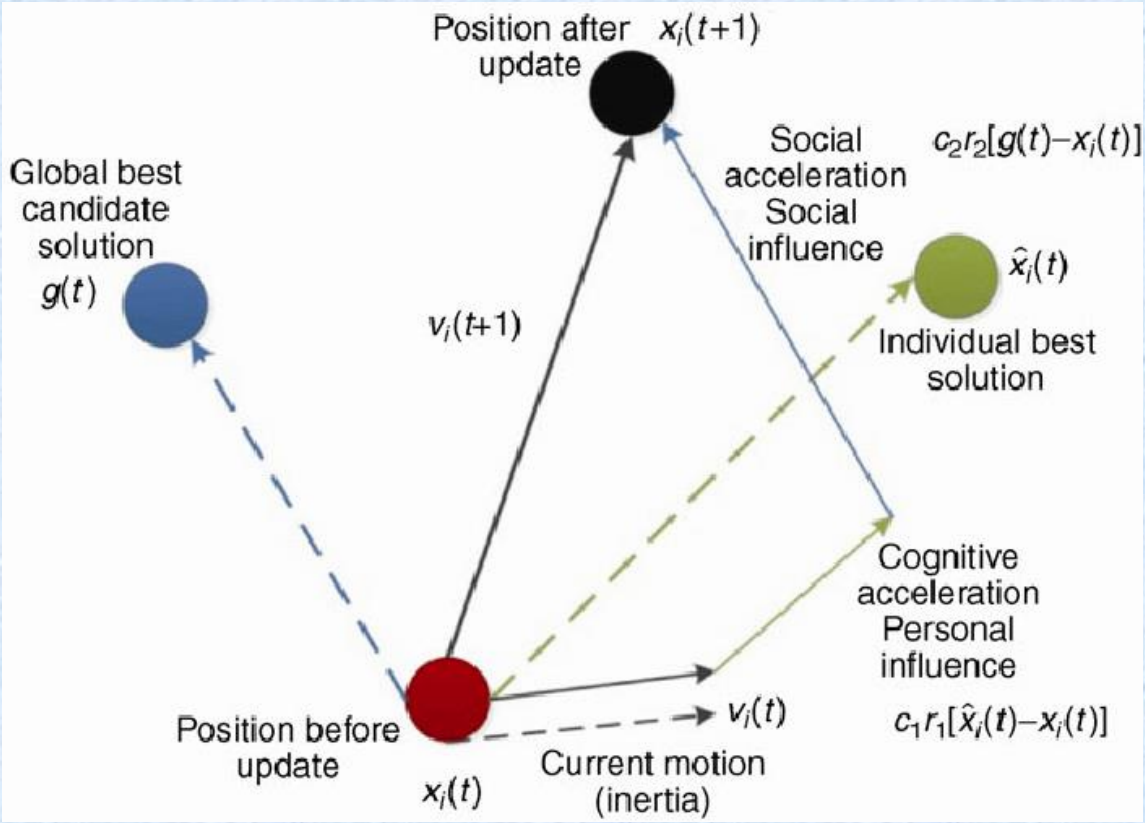
Если ($f(x_i) < f(p_i)$), то делать:

Обновить лучшее известное положение частицы: $p_i \leftarrow x_i$.

Если ($f(p_i) < f(g)$), то обновить лучшее известное состояние роя в целом: $g \leftarrow p_i$.

Теперь g содержит лучшее из найденных решений.

Particle Swarm Optimization (PSO)



Итерационный процесс алгоритма метаэвристического программирования с экспрессией генов

Первая итерация: порождение начальной популяции. Все особи популяции создаются с помощью случайного оператора, порождая значения для каждого элемента вектора в начальной популяции равномерно распределенными от ***0*** до ***1***, с последующим вычислением целевой функции, с проверкой и отсеиванием всех непригодных особей.

Промежуточная итерация: шаг от текущей к следующей популяции. Основной шаг алгоритма состоит в создании нового поколения особей на основе текущей популяции, используя операции миграции особей (ГА - селекции, мутации, кроссовер). При этом происходит декодирование генотипа в фенотип для интерпретации хромосомы как программы для вычисления фитнес-функции.

Последняя итерация (критерий остановки): алгоритм завершается, когда найден вектор с $FF < \varepsilon$, или после заданного числа итераций (генераций) t.

Экспериментальные результаты

Тестовые функции:

№	Определение функции	Число *n*	Диапазон
f1	$f1(x) = x^4 + x^3 + x^2 + x$	1	[1,10]
f2	$f2(x) = \sin(x^2 + x^4)$	1	[1,10]
f3	$f3(x) = \sin(\exp(\sin(\exp(\sin(x)))))$	1	[1,10]
f4	$f4(x) = \sin(x^3) + e^x$	1	[1,10]
f5	$f5(x) = x^5 - 2x^3 + x$	1	[1,10]
f6	$f6(x) = \sin(x) + \sin(x^2 + x)$	1	[1,10]
f7	$f7(x) = \sin(x^2)\exp(x) - 1$	1	[1,10]
f8	$f8(x, y) = 2\sin(x)\exp(y)$	2	[1,10]
f9	$f9(x, y) = \sin(x) + \sin(y^2)$	2	[1,10]
f10	$f10(x, y) = x^3 + y * x^2 + y$	2	[1,10]

В экспериментах использовались значения каждой функции в 10-ти случайных точках в диапазоне (1;10), множество базовых функций , $F_1 = \{+, -, *, sin, exp\}$, терминальные символы для функций Test1-Tes7 - $\{x\}$, остальные $\{x,y\}$.

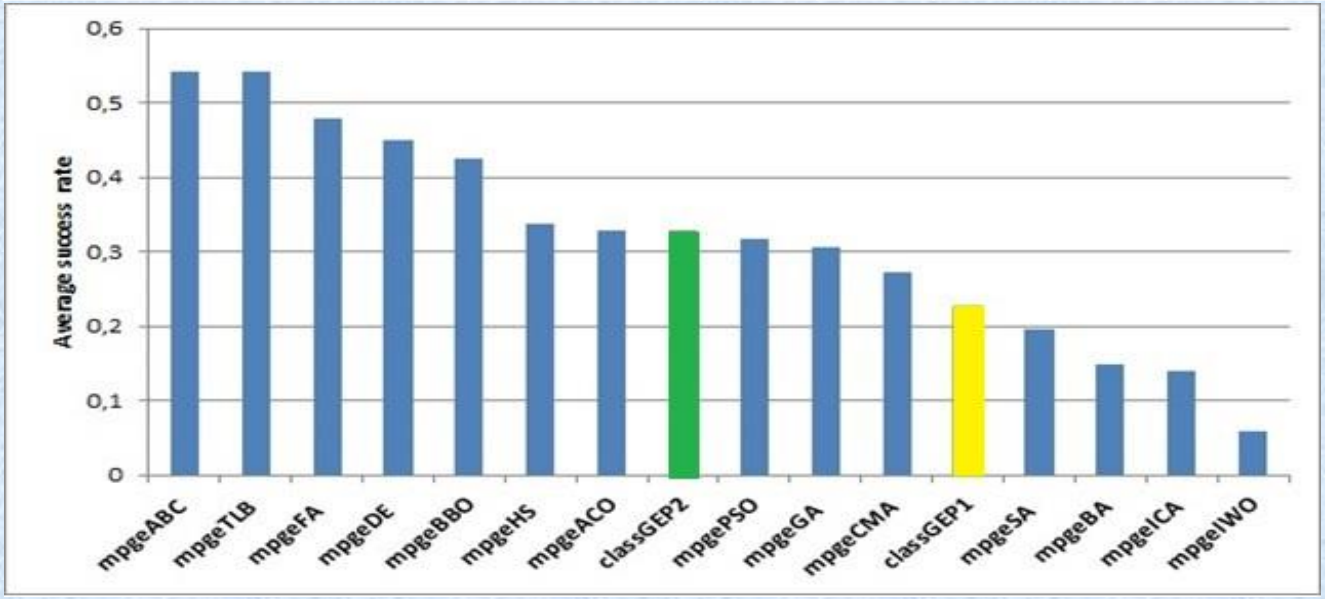
Использовались следующие параметры алгоритмов: размер популяции 200, вероятность кроссовера 0,80, вероятность мутации 0,15, максимальное число генераций 200. Длина хромосом равна 40. Для GEP мы использовали два варианта параметров: размер популяции 200, максимальное количество поколений 200 (для *classGEP1*), размер популяции 1200 и максимальное количество поколений 600 (для *classGEP2*).

В экспериментах использовались следующие четырнадцать биоинспирированных алгоритмов оптимизации: (1) Artificial Bee Colony (ABC), (2) Continuous Ant Colony Optimization (ACOR), (3) Bees Algorithm (BA), (4) Biogeography-based Optimization (BBO), (5) Covariance Matrix Adaptation Evolution Strategy (CMA-ES), (6) differential evolution (DE), (7) Firefly Algorithm (FA), (8) genetic algorithm (GA), (9) Harmony Search (HS), (10) Imperialist Competitive Algorithm (ICA), (11) Invasive Weed Optimization (IWO), (12) Particle Swarm Optimization (PSO), (13) Simulated Annealing (SA), (14) Teaching-Learning based Optimization (TLB).

Программы написаны на языке системы Matlab, эксперименты выполнялись на процессорах Intel Xeon X5670, 2,93 ГГц (Westmere).

Экспериментальные результаты: Частота успешного поиска тестовых функций

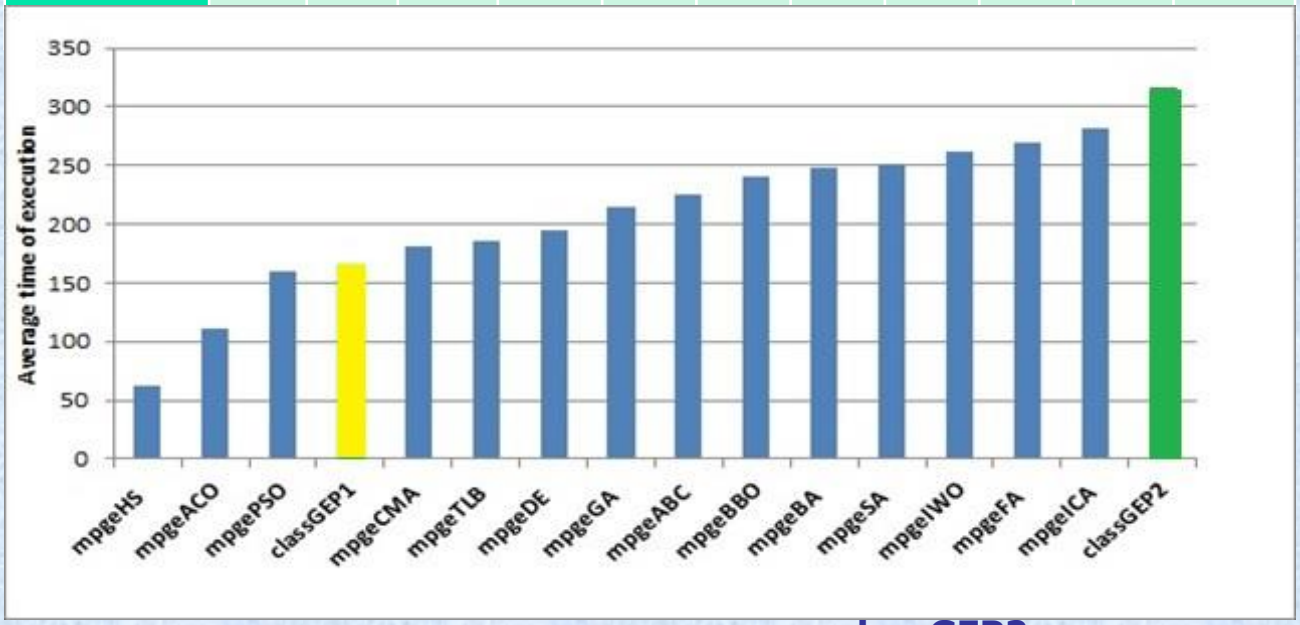
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	$\bar{p}$
mpgeABC	0,63	0,08	0,96	0,33	0,08	0,63	0,38	0,79	0,92	1	0,542
mpgeTLB	0,67	0,08	1	0,08	0	0,75	0,54	0,96	0,75	1	0,541
mpgeFA	0,17	0,04	1	0,38	0,13	0,46	0,13	0,63	1	1	0,479
mpgeDE	0,67	0	1	0,04	0	0,42	0,42	0,75	0,5	1	0,45
mpgeBBO	0,92	0,04	0,88	0,08	0	0,2	0,38	0,79	0,33	1	0,425
mpgeHS	0,42	0,08	0,96	0,04	0	0,2	0	0,33	0,33	1	0,337
mpgeACO	0,5	0	0,5	0,04	0	0,33	0,17	0,65	0,46	0,96	0,328
classGEP2	0.13	0	1	0,25	0	0,21	0,04	0,33	0,67	0,46	0,329
mpgePSO	0,33	0,04	0,97	0	0	0,21	0,13	0,38	0,38	0,79	0,317
mpgeGA	0,63	0	0,45	0,04	0,29	0,17	0,25	0,75	0,13	0,79	0,306
mpgeCMA	0,42	0	0,96	0,13	0	0	0,08	0,54	0,33	0,33	0,273
classGEP1	0	0	0,96	0,17	0	0,08	0,04	0,29	0,33	0,41	0,228
mpgeSA	0,38	0	0,42	0	0	0,04	0,17	0,29	0	0,46	0,195
mpgeBA	0	0	0,79	0,04	0	0,08	0	0,04	0,29	0,04	0,15
mpgeICA	0,08	0	0,17	0	0	0,04	0	0,25	0,08	0,88	0,139
mpgeIWO	0	0	0,21	0	0	0	0	0,08	0,04	0,25	0,058



Вероятность успеха у МП с ЭГ в большинстве случаев выше, чем у классического GEP алгоритма, объясняется способностью МП представлять несколько выражений на одной хромосоме.

Экспериментальные результаты: Среднее время поиска тестовых функций

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	$\bar{t}$
mpgeDE	204	295	24	265	289	233	214	145	167	66.7	195.3
mpgeGA	164	298	195	232	263	258	231	81	214	70.6	214
mpgePSO	201	203	36	174	232	195	209	150	131	58.9	160
mpgeTLB	229	278	7.8	287	348	125	236	65	99	55	185
mpgeSA	202	303	248	250	310	306	268	211	301	164	251.3
mpgeIWO	280	310	256	287	275	301	297	207	209	192	261.4
mpgeICA	88	381	334	307	379	330	372	267	264	80	281.6
mpgeHS	77	73	26	71	84	62	90	57	54	25	62
mpgeFA	314	323	199	293	332	293	334	220	156	186	270
mpgeBBO	79	424	82	369	400	329	308	100	156	19	240.7
mpgeBA	286	273	139	259	280	266	295	201	184	198	247.8
mpgeACO	114	158	9.19	149	165	126	140	97	105	34	111
mpgeABC	264	326	79	285	340	223	314	157	112	41	225.4
mpgeCMA	216	195	13.7	236	332	164	267	92	117	115	202
classGEP1	202	178	46.7	161	211	164	284	172	115	129	166.3



МП с ЭГ имеет меньше времени на поиск решения, чем у classGEP2 во всех случаях (чем у classGEP1 в трех случаях), что объясняется простой структурой данных и операций в МП.

Экспериментальные результаты 2

Мы исследовали эффективность метаэвристического программирования для эволюционного синтеза при использовании трех различных биоинспирированных алгоритмов (генетический алгоритм (GA), дифференциальной эволюции (DE), алгоритм оптимизации роем частиц (PSO)) для задачи поиска аналитического описания модели на основе заданных экспериментальных данных, множества переменных, базовых функций и операций. Были использованы три системы обыкновенных дифференциальных уравнений (ОДУ) с неизвестными функциями и параметрами. Программы написаны на языке системы Matlab, эксперименты выполнялись на процессорах Intel Core i5-8265U, 1.8 GHz с памятью 8 GB.

В экспериментах использовались множество базовых функций для синтеза формул $F = \{+, -, *\}$, терминальные символы - $T = \{x, y, z, C\}$, где $C \in \{\alpha, \beta, \gamma, \delta, a, b, c\}$ – случайные константы. Использовались следующие параметры алгоритмов: размер популяции 50, максимальное число генераций 100, для GA - вероятность кроссовера 0.80, вероятность мутации 0.15, для DE - вероятность кроссовера 0.2, коэффициент масштабирования 0.8, для PSO - w = 0.72984, $c1$ = 1.4962, $c2$ = 1.4962. Длина хромосом равна 90 элементов.

Осциллятор Ван дер Поля

Рассмотрим поиск уравнения осциллятора Ван дер Поля для двумерного случая:

$$\frac{dx}{dt} = y,\ \frac{dy}{dt} = \alpha(1 - x^2)y - x,$$

Пусть неизвестны функции $T_i(x, y)$ в правой части второго уравнения, как показано в : *(1)* $\frac{dy}{dt} = T_1(x, y)y - x$, *(2)* $\frac{dy}{dt} = T_2(x, y) - x$, *(3)* $\frac{dy}{dt} = T_3(x, y)$.

Пусть заданы 50 пар вход – выход $((x, y), \frac{dy}{dt})$ и темплейты *TM* для правой части второго уравнения. Использован алгоритм мета-эвристического программирования для синтеза нелинейных моделей на основе трех БИА (GA, DE, PSO). Пусть t_A - время поиска решения (sec) или исполнения заданного числа поколений при использовании алгоритма *A* и p_A - вероятность (частота) успешного поиска (по результатам 10 экспериментов).

Table 1. Experimental results for the Van der Pol oscillator with the unknown functions T_i.

i	TM	T	t_{DE}	p_{DE}	t_{GA}	p_{GA}	t_{PSO}	p_{PSO}
1	$\frac{dy}{dt} = T_1(x, y)y - x$	$T_1 = \alpha(1 - x^2)$	85	1	113	1	107	1
2	$\frac{dy}{dt} = T_2(x, y) - x$	$T_2 = \alpha(1 - x^2)y$	486	1	686	0.75	1279	0.25
3	$\frac{dy}{dt} = T_3(x, y)$	$T_3 = \alpha(1 - x^2)y - x$	1911	0.25	1866	0.25	2102	0

Модель Ло́тки - Вольте́рры

Рассмотрим поиск уравнения для модели Ло́тки - Вольте́рры взаимодействия двух видов типа «хищник-жертва»:

$$\frac{dx}{dt} = x(\gamma - \delta y), \quad \frac{dy}{dt} = y(\alpha - \beta x),$$

Пусть неизвестны функции $T_i(x, y)$ в правой части второго уравнения, как показано в : $(1)\frac{dy}{dt} = T_1(x, y)y$, $(2)\frac{dy}{dt} = T_2(x, y)$

Пусть заданы 100 пар вход – выход $((x, y), \frac{dy}{dt})$ и темплейты *TM* для правой части второго уравнения. Использован алгоритм мета-эвристического программирования для синтеза нелинейных моделей на основе трех БИА (GA, DE, PSO). Пусть t_A - время поиска решения (sec) или исполнения заданного числа поколений при использовании алгоритма *A* и p_A - вероятность (частота) успешного поиска (по результатам 10 экспериментов).

Table 2. Experimental results for the Lotka - Volterra model with the unknown functions T_i.

i	TM	T	t_{DE}	p_{DE}	t_{GA}	p_{GA}	t_{PSO}	p_{PSO}
1	$\frac{dy}{dt} = T_1(x, y)y$	$T_1 = (\alpha - \beta x)$	2853	0.5	1748	0.5	3292	0.25
2	$\frac{dy}{dt} = T_2(x, y)$	$T_2 = y(\alpha - \beta x)$	4607	0	3772	0.25	4566	0

Система (аттрактор) Лоренца

Рассмотрим поиск уравнения для системы (аттрактора) Лоренца из трех ОДУ, предложенной как простейшая модель атмосферной конвекции:

$$\frac{dx}{dt} = a(y - x), \qquad \frac{dy}{dt} = bx - y - xz, \qquad \frac{dz}{dt} = xy - cz,$$

Пусть неизвестны функции $T_i(x, y, z)$ в правой части третьего уравнения:

$$(1)\frac{dz}{dt} = T_1(x, y, z) - cz, \quad (2)\frac{dz}{dt} = xy + T_2(x, y, z), \quad (3)\frac{dz}{dt} = T_3(x, y, z)$$

Пусть заданы 1000 пар вход – выход $\left((x, y, z), \frac{dz}{dt}\right)$ и темплейты *TM* для правой части третьего уравнения. Использован алгоритм мета-эвристического программирования для синтеза нелинейных моделей на основе трех БИА (GA, DE, PSO).

Table 3. Experimental results for the Lorenz system with the unknown functions T_i.

i	TM	T	t_{DE}	p_{DE}	t_{GA}	p_{GA}	t_{PSO}	p_{PSO}
1	$\frac{dz}{dt} = T_1(x, y, z) - cz$	$T_1 = xy$	383	1	389	1	391	1
2	$\frac{dz}{dt} = xy + T_2(x, y, z)$	$T_2 = -cz$	1596	1	1547	1	1186	1
3	$\frac{dz}{dt} = T_3(x, y, z)$	$T_3 = xy - cz$	32528	0.25	27082	0.25	26920	0.25

1. Вероятность (частота) успешного поиска немного выше у DE, чем у других алг.
2. Среднее время поиска решения не имеет абсолютного лидера.
3. Более высокая специализация темплейта значительно сокращает время поиска.

Синтез нелинейных моделей на основе дифференциальных уравнений в частных производных (на стадии реализации)

Уравнение Клеро: $NT = \{+, -, *, /\}$, $TS = \{x, u, y, Cr, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial x}\}$,

$$u(x,y) = x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + Z(y,x,u) \Rightarrow u(x,y) = x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + sin(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y})$$

Уравнение Кортевéга - де Фри́за: $NT = \{+, -, *, sin\}$, $TS = \{x, u, t, Cr, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}, \frac{\partial^3 u}{\partial x^3}\}$

$$\frac{\partial u}{\partial t} + Z(u,x,t) = 0 \Rightarrow \frac{\partial u}{\partial t} + 6u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0$$

Волновое уравнение: $NT = \{+, -, *, \sin\}$, $TS = \{x, u, t, Cr\}$,

$$\frac{\partial^2 u}{\partial t^2} = \frac{1}{4}\frac{\partial^2 u}{\partial x^2} + Z(u,x,t) \Rightarrow \frac{\partial^2 u}{\partial t^2} = \frac{1}{4}\frac{\partial^2 u}{\partial x^2} + 10\sin(t)u(x,t)$$

$$u(x,0) = \sin(\pi x),\ u(0,t) = 0,\ u(1,t) = 0,$$

Заключение

Рассмотрен подход к решению проблемы построения нелинейных моделей (математических выражений, функций, алгоритмов, программ) на основе заданных экспериментальных данных, множества переменных, базовых функций и операций.

Предложен алгоритм *метаэвристического программирования с экспрессией генов* для синтеза нелинейных моделей –

(1) унифицированный подход, который позволяет использовать для поиска оптимальной модели множество различных метаэвристических алгоритмов,
(2) имеет простое векторное (линейное) представление хромосомы,
(3) простые операции при декодировании генотипа в фенотип для интерпретации хромосомы как математического выражения,
(4) многовариантный метод для представления множества моделей (выражений) с помощью одной хромосомы,
(5) корректность потомков после случайных операторов,
(6) отсутствие эффекта неоправданного роста выражений.

Предложенный подход МП с экспрессией генов был реализован с использованием четырнадцати различных алгоритмов, которые сравнивались друг с другом и со стандартным алгоритмом программирования экспрессии генов (GEP). Было экспериментально исследовано влияние степени специализации темплейта модели на показатели эффективности метаэвристических алгоритмов поиска при синтезе моделей. Эксперименты показывают, что мы всегда можем выбрать наиболее подходящий алгоритм, лучше, чем алгоритм GEP, на основе предпочтительного критерия - либо по времени нахождения решения, либо по вероятности нахождения функции (модели), либо по обоим критериям одновременно.

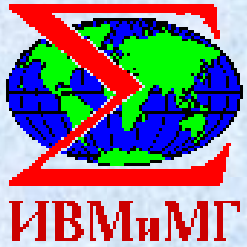
Синтез алгоритмов (экспериментальные результаты)

- вычисления степени и факториала натурального числа,
- определения минимального (максимального) элемента массива,
- определения суммы квадратов элементов массива,
- вычисления скалярного произведения векторов,
- определение формул последовательностей Фибоначчи и Трибоначчи,
- вычисления суммы двух матриц,
- алгоритмов сортировки методами пузырька, слияния и Шелла,
- алгоритмов вычисления корней квадр. уравнений,
- параллельных алгоритмов балансировки нагрузки в многопроцессорных системах,
- алгоритмов определения минимального покрывающего дерева и кратчайших путей в графе
- аналитических описаний новых плотных семейств оптимальных регулярных сетей ,
- нового алгоритма определения функции расстояния для циркулянтных графов степени 4

Публикации:

1. Монахов О.Г., Монахова Э. А. Разработка метода метаэвристического программирования для синтеза нелинейных моделей // Сибирский журнал вычислительной математики, № 4, 2020, с.415-429.
2. O. G. Monakhov, E. A. Monakhova, Comparative Evaluation of Algorithms for Automatic Construction of Nonlinear Models Based on Metaheuristic Programming with Gene Expression // CEUR Workshop Proceedings, 2021, 2965, pp. 254–259.
3. Monakhov O., Evolutionary synthesis of nonlinear models based on metaheuristic programming and templates.// Journal of Physics: Conference Series, Vol. 1715, (MSR-2020), 2021, 012010, 6 p.
4. Monakhov O.G., Monakhova E.A. A parallel algorithm of multi-variant evolutionary synthesis of nonlinear models // Numerical Analysis and Applications. 2017. Т. 10. № 2. С. 140-148.
5. Monakhov O.G., Monakhova E.A., Pant M. Application Of Differential Evolution Algorithm For Optimization Of Strategies Based On Financial Time Series // Numerical Analysis and Applications. 2016. № 2. С. 150-158.
6. Zaheer, H., Pant, M., Kumar, S., Monakhov, O., Monakhova, E., Deep, K. A new guiding force strategy for differential evolution. // International Journal of System Assurance Engineering and Management, vol.8, Suppl.4, 2017, pp. 2170–2183.
7. Monakhov, O., Monakhova, E., A Comparative Analysis of Bioinspired Algorithms for Solving the Problem of Optimization of Circulant and Hypercirculant Networks //Proceedings 2019 15th International Asian School-Seminar Optimization Problems of Complex Systems (OPCS), Novosibirsk, Russia, pp.100-103
8. Monakhov, O., Monakhova, E., An Algorithm of Multi-Variant Evolutionary Synthesis of Nonlinear Models with real-valued chromosomes.//In: Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation. Springer, 2019. pp.41-49.

Thank you for attention



monakhov@rav.sscc.ru