РЕАЛИЗАЦИЯ МЕТАЭВРИСТИЧЕСКИХ АЛГОРИТМОВ В ВИП «ОПТИМУС»

А. Б. Кондратьев, О. В. Коваленко, И. А. Крючков, Д. В. Ежов, А. В. Огородников

ФГУП «РФЯЦ-ВНИИЭФ», г. Саров Нижегородской обл.

Введение

За последние годы в различных государственных органах и ведомствах, а также коммерческих организациях возникла потребность в различного рода средствах поддержки принятия решений, основанных на моделировании и оптимизации поведения действующих или вновь реализуемых систем. Основной задачей этих средств является поиск оптимальных способов функционирования, управления и достижения запланированных результатов в различных подразделениях организации или на предприятии в целом.

Разработанная во ФГУП «РФЯЦ-ВНИИЭФ» визуализационно-интеграционная платформа для оптимизационного имитационного моделирования и управления системами (ВИП «ОптИМУС» [8]) позволяет в кратчайшие сроки реализовать тот или иной программный комплекс для различных заказчиков. Эта платформа является основой для коммерческих приложений с различными реализациями функциональности и пользовательских интерфейсов.

Главными особенностями ВИП «ОптИМУС» являются:

– единый комплекс различных компонентов, обеспечивающий удобный интерфейс для конечного пользователя;

– набор средств для моделирования необходимых задач;

– реализация оптимизационных алгоритмов для заданных параметров;

– привязка к различным системам геопозиционирования;

– различные способы хранения полученных результатов.

Одним из основных блоков платформы является блок оптимизации, который позволяет в автоматическом или полуавтоматическом режиме находить наиболее оптимальное решение задачи в зависимости от заданных параметров и критериев. Найденные оптимальные параметры процесса позволяют не только минимизировать расход материальных ресурсов, но и решить множество других проблем. Например, минимизировать человеческие трудозатраты.

В данной работе рассмотрены реализованные в платформе методы оптимизации, основанные на метаэвристических алгоритмах, а также проведено сравнение эффективности работы алгоритмов на тестовых функциях.

Метавристический подход к нахождению оптимального решения

Метаэвристические алгоритмы – алгоритмы решения задачи, основанные на различных эвристиках, как правило, взятых из природных аналогов или практической деятельности человека. Точность этих алгоритмов не доказано, но практика показывает их выдающиеся результаты по сравнению с так называемыми «точными» алгоритмами.

Метаэвристические алгоритмы применяются для задач высокой вычислительной сложности (задачи класса NP).

Эти алгоритмы применяются в областях искусственного интеллекта, таких как распознавание образов и речи, в компьютерных играх и антивирусных программах и ряде других областях.

Основной критерий эффективности метаэвристических алгоритмов является затраченное время на нахождение оптимального решения как функция размерности задачи. С увеличением размерности моделируемой системы растет и вычислительная сложность решаемой задачи. Размерность задач, которые способны решать метаэвристические алгоритмы за разумное время, в сотни раз выше, чем размерность задач, доступная для «точных» методов оптимизации.

Реализованные оптимизационные алгоритмы в ВИП «ОптИМУС»

В ходе создания платформы были реализованы следующие алгоритмы оптимизации:

– метод Монте-Карло;

– полный перебор в заданном диапазоне;

– метод роя частиц;

– муравьиный алгоритм.

Все вышеперечисленные алгоритмы применяются для определенного рода задач, например, полный перебор не работает в поле непрерывных чисел. Так же, работа каждого алгоритма требует различный объем вычислительных ресурсов.

Постановка задачи оптимизации

Для любой задачи оптимизации в ВИП «ОптИМУС» ставится цель найти минимум или максимум значения целевой функции. Целевой функцией могут выступать любые функции, которые зависят от ко-

нечного числа параметров. Эти параметры могут быть представлены в виде заданных промежутков или набора дискретных переменных.

В качестве тестовой задачи в ВИП «ОптИМУС» была представлена задача «Зомби». Суть задачи состоит в том, что на некотором заданном пространстве располагаются объекты типа «Человек» и «Зомби». Задача «Зомби» заразить как можно большее число людей, а задача людей как можно эффективней убежать от них. Целевой функцией здесь выступило значение «Средней жизни людей». Варьируемым параметром была скорость «Зомби».

Метод Монте-Карло

Метод Монте-Карло является численным методом для решения математических задач при помощи моделирования случайных величин. Данный метод можно применить в большом круге областей: в промышленности для моделирования различных изменяющихся процессов; в физике, в химии, в материаловедении для моделирования различных явлений и поиска оптимальных составов; в области игр для моделирования искусственного интеллекта; в финансовой области для вычисления финансовых инструментов. Метод Монте-Карло применим, практически, в любой области.

При решении задачи оптимизации, метод заключается в том, что для целевой функции случайным образом генерируются набор значений, на основе которых вычисляется значение заданной функции. Лучшее значение по заданному критерию считается оптимальным.

Алгоритм поиска состоит в следующем:

1) задаются границы для параметров задачи и максимальное количество итераций;

2) с помощью независимого распределения необходимо выбрать значения параметров и рассчитать значение целевой функции;

3) если значение целевой функции удовлетворяет условию оптимизации, то его необходимо запомнить;

4) выполнить еще одну итерацию и сравнить полученное значение целевой функции с ранее рассчитанной функцией. Запомнить удовлетворяющее значение;

5) повторять с пункта 2 до тех пор, пока не будет выполнено условие по ограничению количества итераций.

Полный перебор

Полный перебор – это математический метод, используемый для нахождения какого-либо решения путем перебора всевозможных решений из заданной области. Этот метод относится к классу методов решения перебора всех вариантов. Любая задача NP класса может быть решена этим способом. Сложность поиска зависит от всего возможного количества решений исходной задачи. Если пространство решений достаточно велико, то на поиск решения могут уйти достаточно большое количество времени (от нескольких часов до нескольких сотен лет).

Ввиду того, что количество всевозможных решений прикладных задач имеет огромное количество всевозможных решений, метод перебора практически не применяется, разве что для взлома шифров. Иногда использование данного метода оправдано, когда комбинация исходных данных мала.

В задаче оптимизации метод перебора используется в случае необходимости провести расчет с небольшим количеством заданных параметров. При этом данный алгоритм будет более эффективен, чем алгоритмы смешанного поиска.

Описание алгоритма:

1) задаются границы для параметров задачи;

2) составляются всевозможные комбинации параметров задачи;

3) выполнить расчет целевой функции с полученными параметрами и выбрать удовлетворяющее задаче оптимизации решение.

Метод роя частиц

Основная идея метода роя частиц (Particle swarm optimization – PSO) состоит в математическом моделировании поведения роя (стаи) животных в поисках пищи [1, 6]. Данный алгоритм представляет собой метаэвристическую оптимизацию. Как только одна особь из роя увидит путь к цели, остальные особи быстро последуют его примеру. Примерами такого поведения является слет птиц к корму, стая рыб и т. п.

Каждый член стаи рассматривается как частица в пространстве со своими координатами и скоростью движения. Частица, двигаясь по пространству, запоминает координаты наилучшего положения, в котором целевая функция достигала необходимого значения. Так же, каждый член стаи обменивается информацией о «хороших» позициях и использует эту информацию для корректировки своего положения и скорости. На каждой итерации, каждая частица использует прошлый опыт для поиска своего решения поставленной задачи.

Для обмена информацией между членами стаи существует несколько способов. Можно использовать расстояние, так как информация от ближайших соседей порой не обладает какой-то ценностью. Оптимальной идеей будет обмениваться информацией со случайными частицами.

Процедура поиска оптимума заканчивается при достижении максимального числа итераций. В качестве оптимального решения используется наилучшее решение среди всех частиц.

Алгоритм поиска оптимального значения методом роя частиц заключается в следующем:

1) задается количество частиц в стае, $Nl_{\min}$, $Nl_{\max}$ – параметры выбора «соседей» в стае, N –

максимальное число итераций, ω – весовой коэффициент инерции, α и β для вычисления скорости;

2) формируется стая. Положить число итераций $k = 0$. Задать позиции частицам используя равномерный закон распределения $x_i^{j,0} = R[a_i, b_i]$ (1). Положить начальные скорости равными нулю $v^{j,0} = 0$ (2);

3) для каждой частицы найти случайное число Nl_j соседей при помощи равномерного распределения на отрезке $[Nl_{\min}, Nl_{\max}]$, среди них найти частицу с номером n_j и позицией $\hat{x}^{n_j,k}$, которая соответствует заданному оптимизационному значению. Наилучшую позицию запомнить как $\hat{x}^{j,k}$;

4) для каждой частицы в стае найти скорость и положение на следующей итерации $v^{j,k+1} = \omega v^{j,k+1} + \alpha r_1 \left(\hat{x}^{j,k} - x^{j,k}\right) + \omega\beta r_2 \left(\hat{x}^{n_j,k} - x^{j,k}\right)$ (3), $x^{j,k+1} = x^{j,k} + v^{j,k+1}$ (4), где $v^{j,k}, x^{j,k}$ – скорость и положение частицы с номером j на k-ой итерации, r_1 и r_2 – случайные параметры, заданные нормальным законом распределения на отрезке [0, 1]. Если $f\left(x^{j,k+1}\right) < f\left(\hat{x}^{j,k}\right)$ (или $f\left(x^{j,k+1}\right) > f\left(\hat{x}^{j,k}\right)$, в зависимости от оптимизационной задачи), то запомнить новую наилучшую позицию $\hat{x}^{j,k+1} = x^{j,k+1}$ (5), $f\left(\hat{x}^{j,k+1}\right) = f\left(x^{j,k+1}\right)$, (6), $k = k + 1$;

5) проверяется условие выхода. Если $k = N$, то процесс закончить, частицу в стае с оптимальным решением взять как результат. Если $k < N$, то перейти к Шагу 3.

Муравьиные алгоритмы

Муравьиные алгоритмы (Ant colony optimization – ACO) одни из эффективных алгоритмов нахождения оптимального значения искомого параметра в системе [7]. Муравьи – социальные насекомые, которые живут в колонии. Их число в одной колонии может достигать до нескольких миллионов особей и может распространяться на обширные территории. Однако колония не имеет какого-либо централизованного управления, а обмен информацией осуществляется через непрямой обмен информацией, благодаря феромону, испускаемого муравьями в процессе своего движения. Чем больше муравьев пройдет по определенному пути, тем больше концентрация феромонов будет на нем. Описанные принципы лежат в основе рассматриваемых алгоритмов.

Суть подхода заключается в моделировании поведения муравьев, добывающие продовольствие от колонии к источнику питания. Данный алгоритм является метаэвристической оптимизацией.

Изначально муравьиные алгоритмы были использованы для приближенного решения задачи коммивояжера и других аналогичных задач поиска оптимальных маршрутов, но со временем они были расширены на пространство непрерывных чисел. Здесь же будут рассмотрены расширения ACO на непрерывные и дискретно-непрерывные области.

Ant colony optimization for continuous domain (ACO-R)

Идеей алгоритма ACO-R является приращение компонентов вектора варьируемых аргументов, полученных на зависимости от «феромонов» вероятностном выборе компонентов [5]. Это приращение достигается заменой дискретного распределения вероятностей непрерывной функцией (функции плотности вероятности) распределения Гаусса. Алгоритм использует ядро Гаусса для взвешенного суммирования нескольких функций плотности вероятности.

Определим ядро Гаусса как сумму одномерных функций Гаусса (рис. 1):

$$G^i(s) = \sum_{l=1}^{k} \omega_l \, g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{\left(s_l^i - \mu_l^i\right)^2}{2\sigma_l^{i\,2}}}, \qquad (7)$$

где k – число функций плотности вероятности, i – измерение, ω_l – весовая функция, μ_l^i – вектор средних значений (вектор математических ожиданий), σ_l^i – вектор дисперсий.

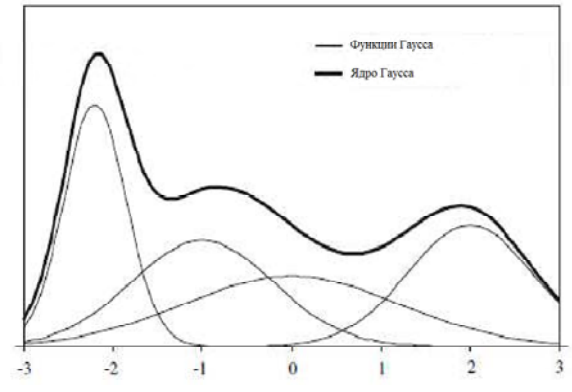

Рис. 1. Функции и ядро Гаусса

Модель «феромонов» ACO-R определяется ранжированием архива решений T (рис. 2). На каждой итерации к архиву решений добавляется N решений и производится упорядочивание с исходным критерием оптимальности. В итоге в архиве будет находиться k решений, в котором худшие решения отсеются. Выше описанная процедура имитирует обновление «феромонов». Целью данной операции является смещение процесса поиска в сторону лучших решений, найденных при оптимизации.

Для удобства описания алгоритма введем массив решений, он же архив решений. В таблице со-

держатся: решения s_l^i согласно их рангу (s_l^i – лучшее решение); ω_l – вес каждой функции плотности вероятности ($\omega_1 \geq \omega_2 \geq \ldots \geq \omega_n$); ядро Гаусса для i-го шага $G^i(s)$.

s_1^1	s_1^2	...	s_1^i	...	s_1^n	ω_1
⋮	⋮	...	⋮	...	⋮	⋮
s_l^1	s_l^2	...	s_l^i	...	s_l^n	ω_l
⋮	⋮	...	⋮	...	⋮	⋮
s_k^1	s_k^2	...	s_k^i	...	s_k^n	ω_k
G^1	G^2	...	G^i	...	G^n	

Рис. 2. Архив решений ACO-R

Для получения решения, муравей на каждом шаге $i = 1 \ldots n$ выбирает значение решения s^i в n-мерной задачи оптимизации.

Сам алгоритм заключается в следующем:

1) для k муравьев случайно генерируются решения $s^i \ldots s^n$;

2) решения упорядочиваются по значениям целевой функции, где $l = 1$ является лучшим рангом;

3) вычисляется вес ω_l для каждого решения, где

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-2)^2}{2q^2k^2}}, \quad (8)$$

где q – коэффициент. При $q \sim 0$ означает, что только функция плотности вероятности лучшего решения была использована для создания нового решения, а при большом q получается более равномерная вероятность;

4) вычисляется вероятность каждого решения

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r}; \quad (9)$$

5) используя метод рулетки, случайно выбирается одно решение s_l, используя рассчитанную вероятность;

6) полагается, что математическое ожидание $\mu_l^i = s_l^i$;

7) вычисляется дисперсия в i-ом измерении по формуле $\sigma_l^i = \xi \sum_{e=1}^{k} \frac{|s_e^i - s_l^i|}{k-1}$ (10), где ξ – коэффициент, определяющий испарение «феромонов»;

8) рассчитывается решение, используя генератор случайных чисел и распределение вероятностей, полученное с помощью ядра Гаусса;

9) вычисляется значение целевой функции;

10) полученные решения добавляются в архив решений T;

11) решения упорядочиваются по значениям целевой функции;

12) k оптимальных решений сохраняются в архиве T;

13) если решение удовлетворяет критериям останова, то поиск завершается, иначе переходят на 3 шаг.

Ant colony optimization for mixed variable (ACO-MV)

ACO-MV представляет собой расширение ACO-R на поле дискретных значений задаваемых параметров [4]. Алгоритм поиска оптимального решения не сильно отличается от алгоритма ACO-R. Главным отличием является добавленный алгоритм поиска оптимального значения для дискретных значений. Архив решений отличается от предыдущего алгоритма только тем, что в нем присутствуют данные по дискретным переменным (рис. 3).

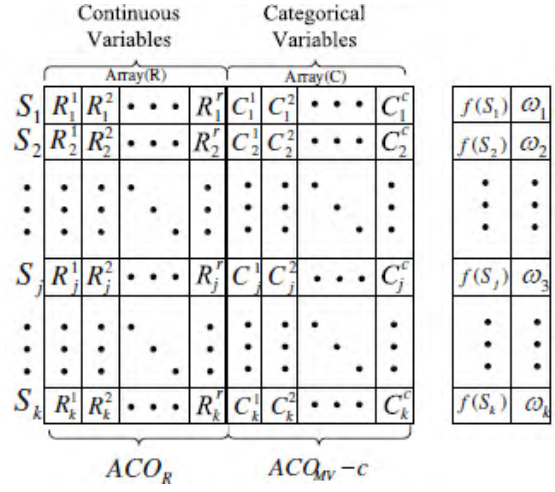

Рис. 3. Архив решений ACO-MV

Измененный алгоритм будет следующим (здесь и далее относится к дискретной части параметров):

1) для k муравьев случайно генерируются решения $s^1 \ldots s^n$;

2) решения упорядочиваются по значениям целевой функции, где $l = 1$ является лучшим рангом;

3) вычисляется вес ω_l для каждого дискретного решения с помощью формулы $\omega_l = \frac{\omega_{jl}}{u_l^i} + \frac{q}{\eta}$ (11), где u_l^i количество решений, использующих v_l^i для i-ой переменной в архиве решений, $v_l^i \in D_i = \left\{ v_l^i, \ldots, v_{c^i}^i \right\}$, η – количество дискретных переменных неиспользованных для поиска оптимума, jl – индекс наилучшего решения, использующий i-ю переменную;

4) вычисляется вероятность каждого решения $o_l^i = \frac{\omega_i}{\sum_{r=1}^{c} \omega_r}$ (12);

5) далее выполняются пункты 5–10 алгоритма ACO-R;

6) после выполняются пункты 3–10 алгоритма ACOR-R для непрерывных параметров;

7) решения упорядочиваются по значениям целевой функции;

8) k оптимальных решений сохраняются в архиве T;

9) если решение удовлетворяет критериям останова, то поиск завершается, иначе переходим на 3 шаг.

Стоит заметить, что при отсутствии дискретных переменных алгоритм ACO-MV становится алгоритмом ACO-R.

Тестовые функции

Для тестирования алгоритмов оптимизации было создано большое количество различных аналитических функций, имеющих множество локальных экстремумов и единственное наименьшее (наибольшее) значение [1]. Основным критерием правильности выполнения алгоритма оптимизации является нахождение глобального наименьшего (наибольшего) значения заданной функции.

В качестве тестовых функций будут рассмотрены следующие функции:

1. функция Растригина:

$$f(x) = 10 + \sum_{i=1}^{d} \left(x_i^2 - 10\cos(2\pi x_i) \right) \qquad (13);$$

2. функция Розенброка:

$$f(x) = \sum_{i=1}^{d-1} \left[\left(1 - x_i\right)^2 + 100\left(x_{i+1} - x_i^2\right)^2 \right] \qquad (14);$$

3. функция Стиблинского–Танга:

$$f(x) = \sum_{i=1}^{d} \left(x_i^4 - 16x_i^2 + 5x_i \right) \qquad (15);$$

4. функция Экли:

$$f(x) = \sum_{i=1}^{d} \left(\left(10^6\right)^{(i-1)/(d-1)} x_i^2 \right) \qquad (16);$$

5. функция Эггхолдера:

$$f(x) = -(x_1 + 47)\sin\sqrt{\left|\frac{x_0}{2} + (x_1 + 47)\right|} - x_0 \sin\sqrt{\left|x_0 - (x_i + 47)\right|}. \qquad (17)$$

Результаты тестирования

В табл. 1 представлены результаты тестирования алгоритмов для различных тестовых функций для случая 5 независимых переменных (функция Эггхолдера зависит от двух переменных).

В табл. 2 представлены результаты тестирования алгоритмов для различных тестовых функций для случая 20 независимых переменных.

Различия в скорости сходимости объясняются зависимостью алгоритмов от их параметров (испарение «феромона», коэффициента инерции и др.). Использование того или иного алгоритма оптимизации позволяет дополнять друг друга в поисках оптимума. В реальных же задачах наперед неизвестны точные значения оптимума целевой функции, поэтому целесообразно применение различных алгоритмов, каждый из которых может привести к «своему» оптимальному (субоптимальному) решению.

Тестовой задачей ВИП «ОПТИМУС» была задача «Зомби» (рис. 4), описанная выше. Целевая функция зависела от одного параметра – скорости агента. Естественно, ограничения по скорости были в тех рамках, чтобы «зомби» не заразили людей в первые секунды жизни.

Таблица 1

Результаты тестирования для $d = 5$

Функция	Монте-Карло	PSO	ACO-R/ACO-MV	Точное значение	Кол-во итераций
Розенброка	2,124	0,038	0,0	0,0	500
Растригина	3,448	0,0	0,0	0,0	500
Стиблинского–Танга	–127,341	–195,831	–195,831	–195,83085	500
Экли	12,922	0,0	0,0	0,0	500
Эггхолдера	–342,179	–959,640663	–959,640663	–959,6407	500

Таблица 2

Результаты тестирования для $d = 20$

Функция	Монте-Карло	PSO	ACO-R/ACO-MV	Точное значение	Кол-во итераций
Розенброка	96,348	72,169	10,639	0,0	500
Растригина	219,401	5,97	70,056	0,0	500
Стиблинского–Танга	–321,448	–784,323	–698,371	–783,3234	500
Экли	48,964	0,0	2,608	0,0	500

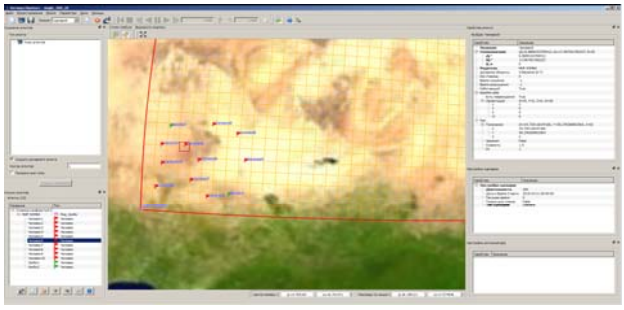

Рис. 4. Начальное положение агентов «зомби» и людей в расчетной области

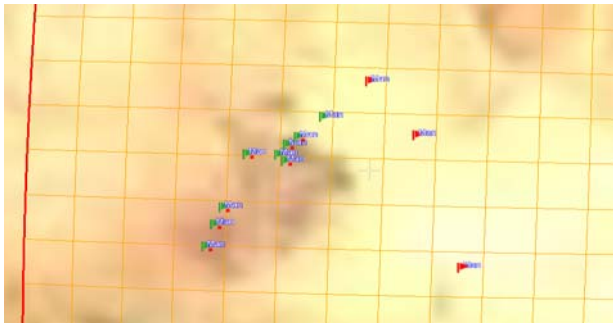

Рис. 5. Движение «зомби» и людей

Проведя расчет, исходя из исходных данных (10 людей, 2 «зомби»), без учета оптимизации средняя продолжительность жизни людей составила 175 секунд модельного времени. При использовании алгоритма оптимизации (PSO), средняя продолжительность жизни уменьшилась до 118 секунд модельного времени.

Заключение

Задачи оптимизации играют большую роль в задачах математического моделирования поведения систем. Используя алгоритмы оптимизации можно сократить дорогостоящие ресурсы компании путем переброса их в другие подсистемы. Метаэвристические алгоритмы позволяют использовать принципы, заложенные природой в математическом моделировании. Порой метаэвристические алгоритмы превосходят в плане сходимости и точности решения точные алгоритмы, а если система не имеет аналитического вида, то метаэвристические алгоритмы будут одним из лучших решений поставленной задачи.

Реализованные в ВИП «ОПТИМУС» алгоритмы оптимизации в дальнейшем будут дорабатываться и совершенствоваться, что может привести к улучшению поиска оптимума. Сейчас эти алгоритмы являются начальным этапом в развитии блока оптимизации на разрабатываемой платформе.

Литература

1. Пантелеев А. В. Метаэвристические алгоритмы поиска глобального экстремума. Москва, 2005.

2. Tianjun Liao, Population-based heuristic algorithms for continuous and mixed discret-continuous optimization problems: Dissertation of doctor of science (engineering). École Polytechnique de Bruxelles, 2012.

3. Zhiming Chen, Shaorui Zhou, Jieting Luo, A robust ant colony optimization for continuous functions // Expert Systems with Applications, 2017. N 81. P. 309–320.

4. Abbas Afshar, Shahrbanoo Madadgar, Ant-Based Algorithms for Optimization of Mixed Variable Domain Engineering Problems; Application to Water Main Pipeline Design.

5. Krzysztof Socha, Marco Dorigo, Ant colony optimization for continuous domains // European Journal of Operational Research, 2008. N 185. P. 1155–1173.

6. Гальченко В. Я., Якимов А. Н. Популяционные метавристические алгоритмы оптимизации роем частиц. Луганск-Черкассы, 2015.

7. Карпенко А. П. Современные алгоритмы поисковой оптимизации. Москва, 2014.

8. Коваленко О. В., Крючков И. А., Ежов Д. В., Огородников А. В., Ерошкина И. В., Собанин Д. С., Хочкин Н. И., Варгина Е. Ф., Тихомиров Ю. В., Рыжих А. В., Васильева Е. А., Кондратьев А. Б. Свидетельство о государственной регистрации программы для ЭВМ № 2019617157 «Визуализационно-интеграционная платформа для оптимизационного имитационного моделирования и управления системами (ВИП «ОПТИМУС»)», правообладатель – ФГУП «РФЯЦ-ВНИИЭФ».