УДК 519.688

Алгоритмическое и программное обеспечение меметического алгоритма поиска условного глобального экстремума

Письменная В.А.

Московский авиационный институт (национальный исследовательский университет), МАИ, Волоколамское шоссе, 4, Москва, А-80, ГСП-3, 125993, Россия

e-mail: wildangel9@yandex.ru

Аннотация

В работе предложен меметический алгоритм поиска условного глобального экстремума функций, основанный на концепции мема, которым является одно из перспективных решений, полученных в ходе реализации процедуры поиска экстремума. На основе предложенного алгоритма сформирован комплекс программных средств на языке C#. Его эффективность продемонстрирована на широко распространённых тестовых задачах поиска глобального условного экстремума функций многих переменных. Кроме того, описано применение данного алгоритма для решения задачи об ориентации космического аппарата.

Ключевые слова: условный глобальный экстремум, мем, меметический алгоритм, оптимизация, гибридный алгоритм, метод имитации отжига, метод муравьиных колоний, оптимальное управление, ориентация космического аппарата.

Введение

На данный момент в математике большое внимание уделяется решению задач глобальной оптимизации. Решение данных задач необходимо в ходе проектирования конструкций летательных аппаратов, когда возникает необходимость оптимизации характерных параметров (вес, дальность полета, аэродинамические характеристики) и разработки систем управления, как отдельными элементами конструкции, так и объектом в целом.

В данной статье рассматривается решение задачи об ориентации космического аппарата (КА). В процессе полёта поддержание ориентации космического аппарата является наиболее распространённой задачей. В статье рассмотрен плоский вариант задачи: поворот КА производится в одной плоскости за счет вращения маховика, установленного внутри КА. В начальный момент времени угловая скорость КА и угол ориентации равны нулю. За счёт вращения маховика необходимо за конечное время привести угол ориентации КА к заданному значению, обнулив при этом угловую скорость [1,2].

Применение разработанного алгоритма целесообразно также для ряда других оптимизационных задач, таких как задача о стабилизации спутника (задача гашения вращательного движения спутника с помощью установленных на нем двигателей) [3], задача о перехвате (перехват ракетой цели), задача о стабилизации высоты летательного аппарата. Применение описанного в статье алгоритма значительно сокращает временные затраты, необходимые для решения подобных задач.

Использование существующих численных методов связано с рядом сложностей: большими вычислительными нагрузками, требованиями к постановке задачи, трудностями в достижении сходимости метода. Ввиду этого выявляется необходимость разрабатывать и использовать так называемые эвристические методы. Данные методы не имеют строгого обоснования сходимости, однако в большинстве случаев они позволяют получать приемлемое решение задачи.

Термин «меметические алгоритмы» (МА) широко используется в качестве обозначения взаимодействия эволюционного или другого подхода, основанного на понятии популяции, и индивидуального обучения особей либо другой локальной процедуры улучшения решения для задач поиска глобального экстремума.

Теория «универсального дарвинизма», предложенная Ричардом Докинзом (Richard Dawkins) [4], полагает, что понятие эволюции применимо не только к биологическим системам, но и к любой сложной системе, которой присущи принципы наследования, изменения и селекции, т.е. все принципы развития. К примеру, наука меметика представляет собой аналог генетики в развитии культуры. Термин «мем» был введён Р.Докинзом как «единица передачи культурной информации, распространяемая от одной особи к другой посредством имитации, научения и др.» [5].

Термин «меметический алгоритм» был впервые предложен П.Москато (P.Moscato) в [6], где он рассматривал МА как гибрид генетического алгоритма и процедуры индивидуального обучения для уточнения решения задачи. На этапе индивидуального обучения решение (особь или её генотип) заменяется новым

(обученным) решением в случае, если новое решение имеет большую приспособленность независимо от остальной части популяции. Таким образом, происходит так называемое культурное развитие особи, которое затем передаётся её потомкам в течение последующих поколений.

В данный момент МА также называют гибридными эволюционными алгоритмами, эволюционными алгоритмами Болдуина (Baldwinian Evolutionary Algorithms), эволюционными алгоритмами Ламарка (Lamarckian Evolutionary Algorithms), культурными алгоритмами или генетическими алгоритмами локального поиска.

К первому поколению МА относятся алгоритмы-гибриды эволюционного подхода, используемого для глобального поиска, и культурной эволюционной составляющей. Хотя это поколение МА и включает в себя характеристики культурной эволюции (в форме локального уточнения решения) в цикле поиска, оно не до конца отвечает требованиям развивающейся системы в соответствии с идеей «универсального дарвинизма», так как все основные принципы: наследование (передача мемов), изменение и селекция отсутствуют. Это объясняет, почему термин МА вызвал критику и споры среди исследователей, когда впервые был предложен [6].

Второе поколение МА – это так называемые мульти-МА [7], гипер-эвристические МА [8] и мета-МА Ламарка [9] (Meta-Lamarckian MA), которые проявляют принципы меметической трансмиссии и селекции в своих реализациях. В мульти-МА меметический материал кодируется как часть генотипа. Далее

декодированный мем каждой особи используется для локального уточнения решения. Меметический материал передаётся с помощью простого механизма наследования от родителей к потомкам. С другой стороны, в гипер-эвристических и мета-МА Ламарка кандидаты в мемы конкурируют между собой на основании их прошлых заслуг в локальном улучшении решения. Конкуренция мемов обеспечивается механизмом вознаграждения лучших мемов, на основании чего принимается решение о выборе того или иного мема для последующего уточнения решения задачи. Мемы с большим количеством наград имеют больше шансов на воспроизведение или копирование. Таким образом, МА второго поколения используют несколько методов индивидуального обучения особей в рамках эволюционной системы [10].

В качестве третьего поколения МА выделяют коэволюцию [11] и самогенерирующиеся МА [12]. Здесь выполнены все три принципа развивающихся систем согласно теории «универсального дарвинизма». В отличие от второго поколения МА, в которых предполагается, что используемые мемы известны изначально, третье поколение МА использует локальный поиск, основанный на некоторых правилах, для пополнения множества кандидатов в решение в рамках эволюционной системы, тем самым учитывая регулярно повторяющиеся структуры и черты данной области задач.

Меметические алгоритмы являются предметом интенсивных научных исследований и успешно применяются для множества реальных задач. Примерами таких задач служат: задача коммивояжёра, задача о назначениях, задача о

минимальной раскраске графа, задача упаковки в контейнеры. Более поздние применения включают в себя обучение искусственных нейронных сетей [13], распознавание образов [14], медицинские экспертные системы [15] и многое другое.

В разработанном алгоритме культурная эволюционная составляющая реализуется в ходе решения подзадачи оптимизации любым из двух методов: с помощью метода муравьиных колоний [16-18] или методом имитации отжига [17,19]. В ходе культурной эволюции информация о мемах используется для генерации более совершенной в терминах решаемой задачи особи.

Постановка задачи

Дана целевая функция $f(x) = f(x_1, x_2, ..., x_n)$, определённая на множестве допустимых решений $D \subseteq \mathbb{R}^n$. Требуется найти условный глобальный минимум функции $f(x)$ на множестве D, т.е. такую точку $x^* \in D$, что

$$f(x^*) = \min_{x \in D} f(x), \tag{1}$$

где $x = (x_1, x_2, ..., x_n)^T$, $D = \{x \mid x_i \in [a_i, b_i], i = 1, 2, ..., n\}$.

Задача поиска максимума функции $f(x)$ сводится к задаче поиска минимума путем замены знака перед функцией на противоположный:

$$f(x^*) = \max_{x \in D} f(x) = -\min_{x \in D}[-f(x)]. \tag{2}$$

Функция $f(x)$ может быть многоэкстремальной, поэтому искомое решение в общем случае неединственное.

Стратегия поиска решения

Поясним основную идею МА и используемые термины. МА относятся к эволюционным метаэвристическим алгоритмам поиска глобального условного экстремума функций многих переменных. В данном алгоритме мемом является одно из перспективных решений, полученных в ходе реализации процедуры поиска экстремума. Мемы хранятся в множестве $Pool$. На основании информации о мемах находятся новые решения, претендующие на место в множестве $Pool$. Множество $Pool$ пополняется мемами вплоть до достижения максимального числа мемов, равного K.

Рассматриваемая целевая функция $f(x)$ эквивалентна природному понятию приспособленности живого организма. Каждый допустимый вектор $x = (x_1, x_2, ..., x_n)^T \in D$ является возможным решением поставленной оптимизационной задачи и называется особью. Генерируемое на каждой итерации алгоритма конечное множество решений (особей) называется популяцией.

Стратегия поиска решения включает процедуры формирования начальной популяции и множества $Pool$, реализацию локального поиска и механизм обновления множества $Pool$.

Сначала формируется начальная популяция, состоящая из m особей $x^1, x^2, ..., x^m$, при помощи равномерного распределения на множестве допустимых решений. Для каждой особи популяции вычисляется значение функции приспособленности (целевой функции $f(x)$), после чего список особей сортируется от особи с наилучшей приспособленностью к особи с наихудшей

приспособленностью. Особь с наилучшей приспособленностью помещается в множество $Pool$. Далее путём последовательного перебора оставшихся особей из отсортированного списка находится наилучшая особь среди находящихся на достаточном расстоянии от особи, уже помещенной в множество $Pool$ (расстояние должно превышать пороговое значение σ). Она тоже помещается в множество $Pool$. Обе отобранные особи считаются мемами (текущее число мемов равно $k = 2$) и обозначаются x^1_{pool}, x^2_{pool}.

Рассмотрим процедуру локального поиска. С использованием мемов из множества $Pool$ решается задача условной минимизации целевой функции, в качестве аргумента которой будет выступать некоторая новая точка x^{new} – претендент в множество $Pool$:

$$f(x^{new}) \to \min_{c_1, c_2, ..., c_k}, \tag{3}$$

где $x^{new} = c_1 x^1_{pool} + c_2 x^2_{pool} + ... + c_k x^k_{pool}$; $x^{new} \in D$; $x^1_{pool}, x^2_{pool}, ..., x^k_{pool} \in Pool$; $c_1, ..., c_k \in C \subseteq R^k$; C – множество допустимых значений коэффициентов $c_1, ..., c_k$; k – текущее значение числа мемов в множестве $Pool$ ($2 \le k \le K$).

Для нахождения наилучших значений коэффициентов $c_1, ..., c_k$ предлагается использовать непрерывный вариант метода муравьиных колоний [16] или метод имитации отжига [17]. Поскольку в стратегии используются сразу несколько эвристических процедур, метод является гибридным.

После решения задачи локального поиска полученная точка x^{new} добавляется в множество $Pool$, и проверяется условие окончания заполнения этого множества.

Если $k = K$, процедура локального поиска, связанная с заполнением множества $Pool$, завершается. Если $k < K$, необходимо снова решить задачу минимизации функции $f(x^{new})$ с числом коэффициентов, большим на единицу, и добавить в множество $Pool$ полученную точку x^{new}.

После завершения заполнения множества $Pool$ принимается решение о продолжении поиска. Если число выполненных итераций локального поиска не достигло максимального числа $M_{\max}$, то происходит обновление множества $Pool$. Сначала среди точек, находящихся в множестве $Pool$, выбирается точка с наилучшей приспособленностью, которая записывается на «лист памяти». Затем из множества $Pool$ удаляются q наихудших решений. Оставшиеся решения проверяются на близость друг к другу, т.е. происходит проверка условия

$$d(x^{j_1}, x^{j_2}) = \sqrt{\sum_{i=1}^{n} (x_i^{j_1} - x_i^{j_2})^2} > \sigma, \qquad (4)$$

где $x^{j_1}, x^{j_2} \in Pool,$ и наихудшее из двух близких решений удаляется из множества $Pool$.

После этого формируется новая популяция так же, как и начальная. Наилучшее решение в этой популяции добавляется в множество $Pool$. Далее реализуется процедура локального поиска.

В случае достижения максимального количества итераций $M_{\max}$ из «листа памяти» выбирается наилучшая точка (с наилучшим значением функции приспособленности), которая считается приближенным решением задачи условной минимизации целевой функции.

Алгоритм

Шаг 1. Задать максимальное число итераций $M_{\max}$, размер популяции m, параметр σ – пороговое значение расстояния между точками, K – максимальное количество элементов множества $Pool$, q – количество удаляемых решений из множества $Pool$ на каждой итерации, область определения C коэффициентов $c_1,...,c_k$, параметры метода муравьиных колоний или имитации отжига в зависимости от сделанного выбора. Положить $k=0$ – количество элементов в множестве $Pool$; $M=0$ – номер итерации.

Шаг 2. Формирование множества $Pool$.

Шаг 2.1. Случайным образом сформировать популяцию I_M. Для этого с помощью равномерного распределения m раз сгенерировать последовательность из n случайных точек $\{P_i^{M,p}\}_{i=1}^n$, $i=1,...,n$, $p=1,2,...,m$ на отрезке $[0,1]$. Используя линейное преобразование, каждая точка отображается на соответствующий ей промежуток $[a_i,b_i]$: $P_i^p=(b_i-a_i)P_i^{M,p}+a_i$. Составляя векторы из точек последовательности $\{P_i^p\}_{i=1}^n$ при фиксированном p, получаем m начальных векторов $x^p=(x_1^p,x_2^p,...,x_n^p)^T$, $x_i^p=P_i^p$, $i=1,2,...,n$, координаты которых x_j^p имеют равномерное распределение на отрезках $[a_i,b_i]$, $i=1,...,n$. Таким образом, может быть сформирована начальная популяция $I_M=\{x^p, p=1,2,...,m \mid x^p=(x_1^p,x_2^p,...,x_n^p)^T\in D\}$.

Шаг 2.2. Вычислить значение функции приспособленности для каждой особи $x^p \in I_M$: $f_p = f(x^p)$, $p = 1,...,m$. В соответствии со значениями функции приспособленности $f(x^p)$ упорядочить векторы x^p от лучшего x^{best} (обеспечивающего наименьшее значение $f(x^p)$) к худшему (обеспечивающему наибольшее значение $f(x^p)$).

Шаг 2.3. Лучшее решение x^{best} поместить в множество $Pool$. Положить $x_{pool}^{k+1} = x^{best}$. Если $M = 0$, то перейти к шагу 2.4. Иначе перейти к шагу 3.

Шаг 2.4. Решение x^j, следующее за x^{best} в упорядоченном списке и удовлетворяющее условию

$$d(x^{best}, x^j) = \sqrt{\sum_{i=1}^{n} (x_i^{best} - x_i^j)^2} > \sigma,$$

поместить в множество $Pool$. Положить $x_{pool}^{k+2} = x^j$. Положить $k = k + 2$. Если такого решения нет, то перейти к шагу 2.1.

Шаг 3. Решение задачи локального поиска.

Шаг 3.1. Решить задачу нахождения нового решения x^{new}:

$$f(x^{new}) \to \min_{c_1, c_2, ..., c_k},$$

где $x^{new} = c_1 x_{pool}^1 + c_2 x_{pool}^2 + ... + c_k x_{pool}^k$; $x^{new} \in D$; $x_{pool}^1, x_{pool}^2, ..., x_{pool}^k \in Pool$; $c_1, ..., c_k \in C$. Для решения этой задачи применить метод муравьиных колоний или метод имитации отжига.

Шаг 3.2. Поместить x^{new} в множество $Pool$. Положить $x_{pool}^{k+1} = x^{new}$, $k = k + 1$.

Шаг 3.3. Если $k < K$, то перейти к шагу 3.1.

Если $k = K$, перейти к шагу 4.

Шаг 4. Обновление множества $Pool$.

Шаг 4.1. Упорядочить решения, находящиеся в множестве $Pool$, в соответствии со значениями функции приспособленности $f(x_{pool}^{j})$, $j = 1,..., K$ от лучшего к худшему. Наилучшее решение записать на «лист памяти».

Шаг 4.2. Положить $M = M + 1$. Если $M < M_{\max}$, то перейти к шагу 4.3. Иначе выбрать наилучшее решение из «листа памяти» и закончить выполнение алгоритма.

Шаг 4.3. Удалить q наихудших решений из множества $Pool$, положить $k = k - q$.

Шаг 4.4. Удалить из множества $Pool$ худшее из двух решений $x_{pool}^{j_1}, x_{pool}^{j_2}$, находящихся слишком близко друг к другу, т.е. не удовлетворяющих условию:

$$d(x_{pool}^{j_1}, x_{pool}^{j_2}) = \sqrt{\sum_{i=1}^{n} (x_{pool\,i}^{j_1} - x_{pool\,i}^{j_2})^2} > \sigma; \quad \forall x_{pool}^{j_1}, x_{pool}^{j_2} \in Pool,$$

каждый раз полагая $k = k - 1$. Перейти к шагу 2.

Замечание. Рекомендуемые значения параметров алгоритма: $M_{\max} = 100 \div 200$; $m = 50 \div 100$; $\sigma = 0,001$; $K = 10 \div 15$; $q = \left[\frac{K}{2}\right]$; $c_k \in [-5;5]$.

Программное обеспечение

На основе предложенного алгоритма автором разработано программное обеспечение для поиска глобального минимума функций. Среда разработки – Microsoft Visual Studio, язык программирования – C#.

С помощью программного обеспечения пользователь может выполнять следующие действия: вводить параметры постановки задачи; задавать параметры метода; проводить серии решений одной и той же задачи с одинаковыми или различными значениями параметров алгоритма; получать статистические данные по проведенной серии решений.

Благодаря графическому отображению процесса решения, в ряде случаев можно сразу отследить, насколько эффективно работает алгоритм.

В программе имеется возможность выбора метода решения задачи локального поиска: муравьиных колоний (непрерывный вариант) либо имитации отжига. Кроме того, пользователь может выбрать одну из стандартных тестовых функций с помощью выпадающего списка в правом верхнем углу окна или осуществить ввод произвольной целевой функции.

На рис. 2 показан общий вид интерфейса программы на примере минимизации функции Розенброка. В качестве метода решения задачи локального поиска выбран метод муравьиных колоний. В верхней части окна указываются значения параметров задачи (целевая функция, границы множества допустимых решений) и параметров метода. В нижней части отображается результат (точка минимума функции, значение функции в точке минимума и время, затраченное на работу

алгоритма в миллисекундах), а также среднее и минимальное значения функции и среднеквадратическое отклонение в результате проведения 100 тестов. На рис. 3 показан вид окна при выборе метода имитации отжига для решения данной задачи.

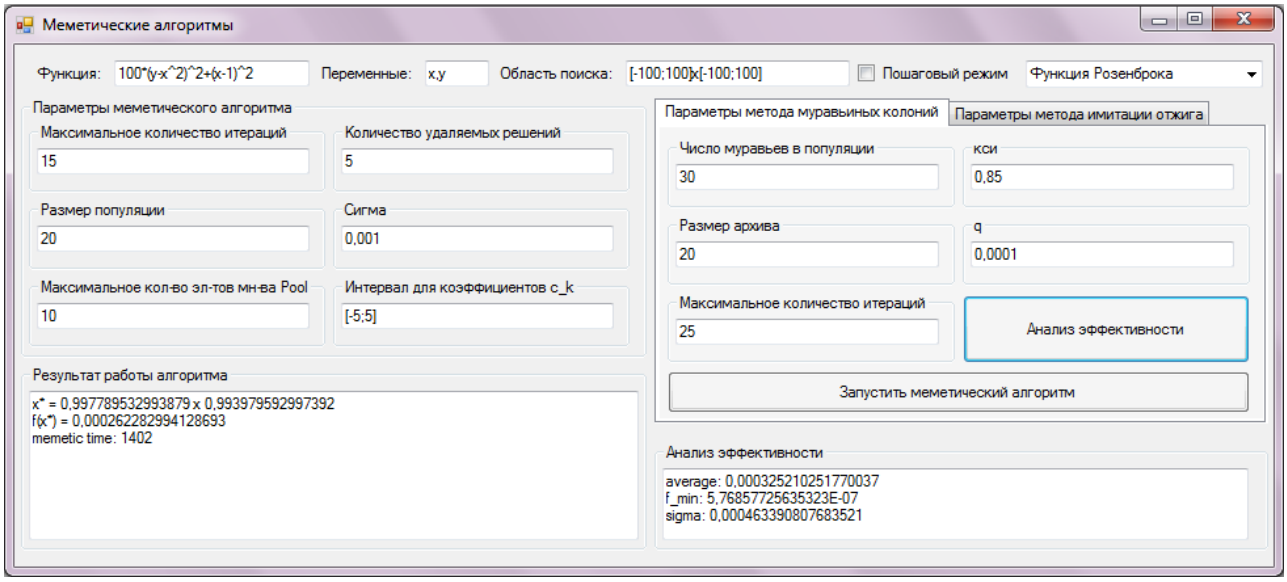

Рис. 1. Окно программы при выборе метода муравьиных колоний в качестве метода решения задачи локального поиска

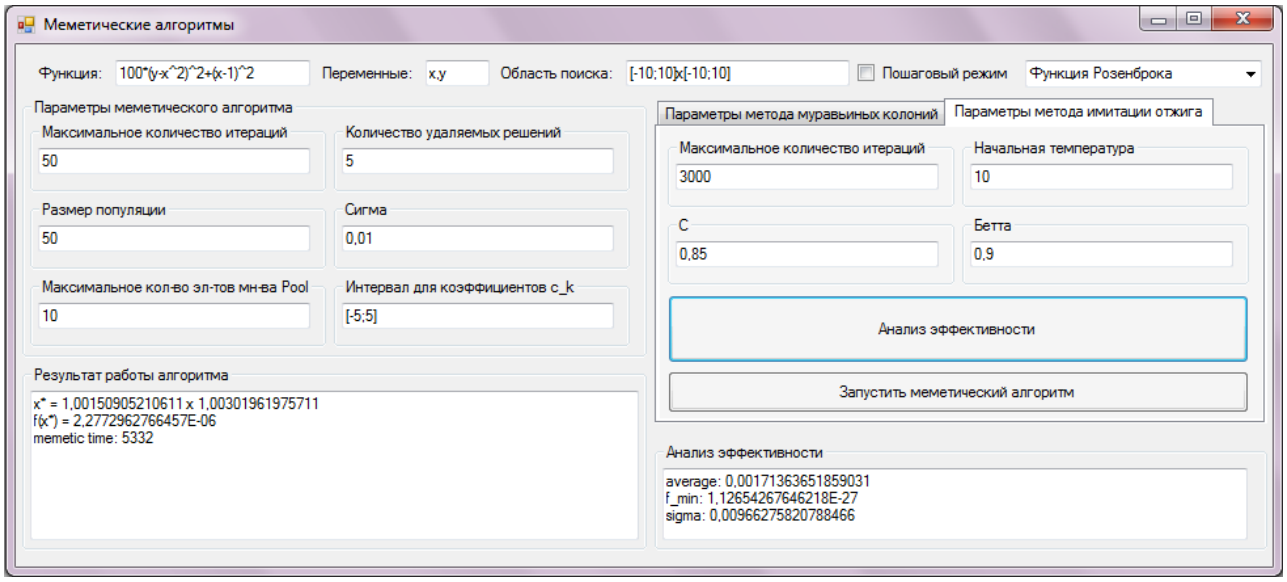

Рис.2. Окно программы при выборе метода имитации отжига в качестве метода решения задачи локального поиска

Кроме того, имеется возможность просмотра пошаговой работы алгоритма, а для функций одной и двух переменных – графического представления результатов

его работы. На рис. 4 показано графическое представление и пошаговый вывод работы метода для функции Розенброка.

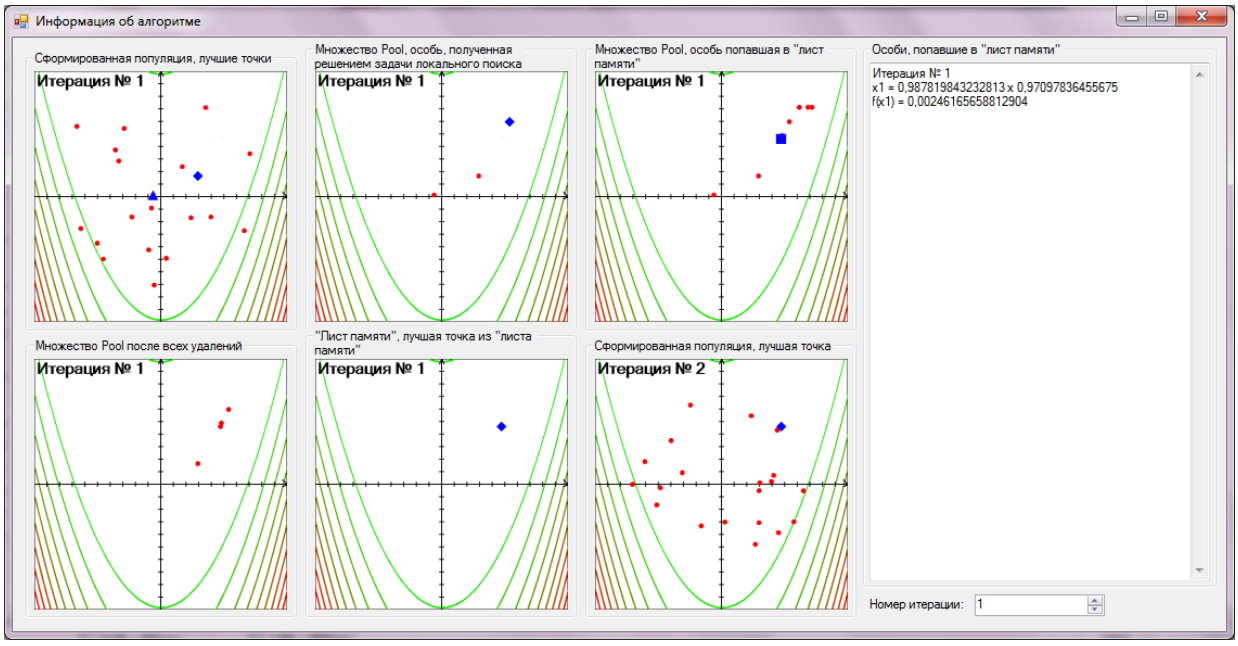

Рис. 3. Окно графического представления работы меметического алгоритма. Первая итерация

На рисунках изображены линии уровня исследуемой функции. На первом рисунке точками изображаются особи сформированной популяции I_M; ромбом – наилучшая особь популяции, попавшая в множество $Pool$; треугольником – особь, следующая за лучшей в упорядоченном списке и находящаяся на расстоянии, большем σ, также попавшая в множество $Pool$. На втором рисунке точками изображаются особи из множества $Pool$, а ромбом – особь, полученная решением задачи локального поиска. На третьем рисунке точками изображаются особи из множества $Pool$ после его заполнения ($k = K$), квадратом выделена лучшая особь, попавшая в «лист памяти». На следующем рисунке изображаются точки, оставшиеся в множестве $Pool$ после всех удалений. На пятом рисунке изображаются особи, находящиеся в «листе памяти» на текущей итерации, лучшая

из них выделяется ромбом. На последнем рисунке изображается сгенерированная популяция на следующей итерации алгоритма, ромбом выделяется точка, попавшая в *Pool*. В правой части окна представлен список особей, попавших в «лист памяти» на каждой из просмотренных итераций, а также значение целевой функции в точках с этими координатами.

На рис. 5 представлена последняя итерация работы алгоритма. Наилучшая точка из «листа памяти» (являющаяся ответом): $x = 0{,}996172623873161; y = 0{,}991837895197196$. Значение функции в этой точке: $f(x, y) = 0{,}00004189735$. Данная точка достаточно близка к точному ответу задачи: $x^* = y^* = 1,\ f(x^*, y^*) = 0$.

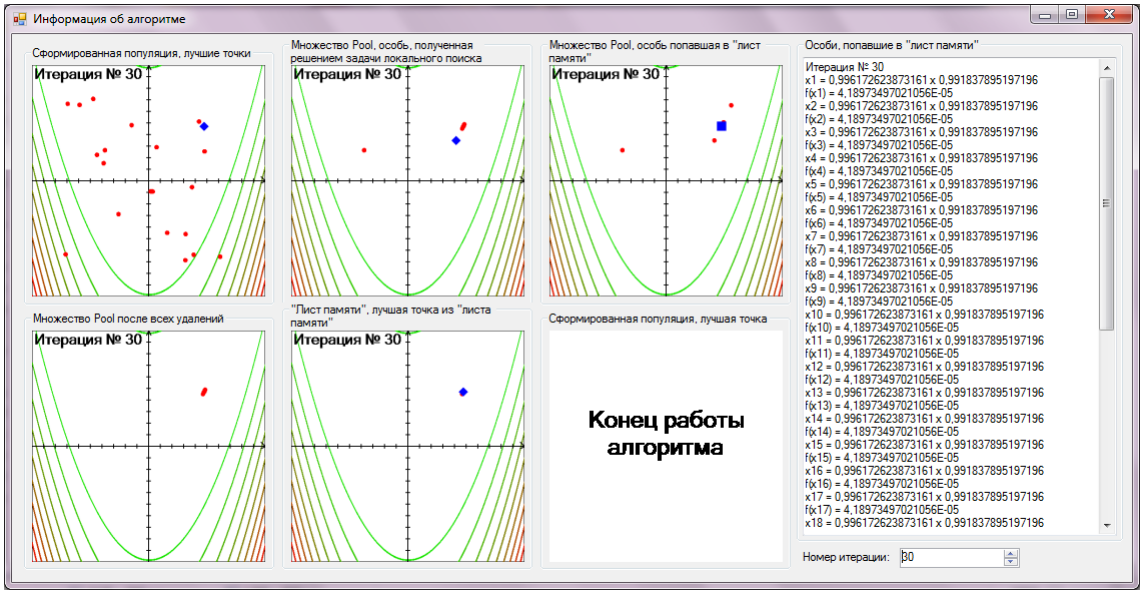

Рис. 4. Окно графического представления работы меметического алгоритма. Последняя итерация

Рекомендации по выбору параметров метода

Максимальное количество итераций $M_{\max}$ определяет, как долго будет продолжаться поиск новых решений. Чем больше $M_{\max}$, тем более точным будет решение, если при этом другие параметры алгоритма подобраны подходящим

образом. Однако при неоправданно больших значениях $M_{\max}$ без необходимости расходуются вычислительные ресурсы.

Чем больше размер популяции m, тем тщательнее исследуется множество допустимых решений на начальных шагах поиска. При этом с увеличением количества особей в популяции время работы алгоритма увеличивается незначительно, но достаточно сильно повышается точность найденного решения.

Максимальное количество K элементов множества $Pool$ существенно увеличивает время работы алгоритма, однако не сильно влияет на точность результатов его работы. В связи с этим не следует выбирать данный параметр слишком большим. Рекомендуемые значения данного параметра: $10 \div 15$.

Количество удаляемых решений q из множества $Pool$. Увеличение количества удаляемых решений влияет на качество исследования множества допустимых решений, а, следовательно, положительно сказывается на точности, но существенно снижает скорость работы алгоритма. При незначительных изменениях количества удаляемых элементов (в сравнении с числом элементов самого множества $Pool$) точность полученного решения практически не изменяется. В связи с этим рекомендуется выбирать количество удаляемых решений, равное приблизительно $\frac{K}{2}$.

Пороговое значение близости между решениями σ влияет на количество удаляемых из множества $Pool$ решений при завершении его заполнения, а также на то, насколько близко будут расположены особи, добавляемые в это множество при его формировании. Большое значение параметра может негативно повлиять на

результат работы алгоритма, так как могут быть удалены особи, потенциально сходящиеся к точке глобального экстремума. При малых значениях параметра происходит незначительное сокращение множества. При этом в популяции могут остаться особи, находящиеся в окрестности одного и того же экстремума.

Интервал для коэффициентов c_k (при выборе метода муравьиных колоний) влияет на качество исследования множества допустимых решений и вероятность отыскания глобального экстремума. Расширение интервала незначительно увеличивает время работы алгоритма. В то же время слишком маленький интервал может не позволить эффективно определить новое решение x^{new}. Рекомендуемый интервал для коэффициентов c_k: $[-5;5]$.

Тестовые примеры. Анализ эффективности метода

Во всех примерах, приведенных ниже, проводились серии из 100 решений одной и той же задачи с одними и теми же значениями параметров. Для полученной выборки $\{f^1, f^2, ..., f^{100}\}$ вычислялись выборочное среднее $\overline{f} = \frac{1}{100}\sum_{i=1}^{100} f^i$ и среднеквадратическое отклонение $\overline{\sigma}_f = \sqrt{\overline{S}_{100}}$, $\overline{S}_{100} = \frac{1}{99}\sum_{i=1}^{100}(f^i - \overline{f})^2$. Для опытов с использованием метода муравьиных колоний $c_k \in [-5;5]$.

Пример 1. Целевая функция (функция Экли):

$$f(x,y) = -20\exp\left(-0,2\sqrt{0,5(x^2+y^2)}\right) - \exp\left(0,5(\cos 2\pi x + \cos 2\pi y)\right) + 20 + e,$$

область поиска $D = [-100:100]\times[-100;100]$. Точное решение: $x^* = y^* = 0$, $f(x^*, y^*) = 0$. Результаты решения примера отражены в табл. 1 и 2.

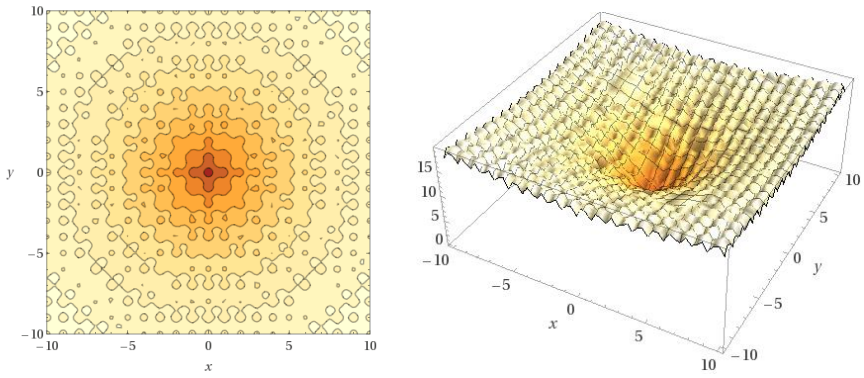

Рис.5. Линии уровня и график функции Экли

Влияние параметров меметического алгоритма (при использовании метода муравьиных колоний). Функция Экли

Таблица 1

Параметры меметического алгоритма					Параметры метода муравьиных колоний					$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	m	r	K	ξ	q			
25	20	10	5	0,001	20	10	10	0,85	0,0001	0,08557	0,00144	0,16113
25	20	10	5	0,1	20	10	10	0,85	0,0001	0,09604	0,0008	0,10876
50	20	10	5	0,1	10	5	20	0,85	0,0001	**0,00146**	**0**	0,00191
25	50	10	5	0,1	10	5	20	0,85	0,0001	**0,00156**	**0**	0,00277
25	30	20	5	0,1	10	5	20	0,85	0,0001	0,14108	**0**	0,51175

Влияние параметров меметического алгоритма (при использовании метода имитации отжига). Функция Экли

Таблица 2

Параметры меметического алгоритма					Параметры метода имитации отжига				$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	N	T_0	C	β			
25	20	10	5	0,001	1000	1000	0,85	0,95	1,57947	0,1972	0,82634
25	20	10	5	0,001	3000	100	0,85	0,95	1,47809	**4,44E-16**	0,84828
25	20	10	5	0,01	3000	10	0,85	0,9	**0,050997**	**4,44E-16**	0,27062
25	20	10	5	0,1	4000	5	0,85	0,9	**3,29E-06**	**4,44E-16**	2,27E-05
25	20	10	5	0,1	4000	5	0,88	0,99	0,087762	**0**	0,398621
30	50	10	5	0,1	2000	5	0,88	0,9	**2,54E-07**	**4,44E-16**	1,93E-06

Пример 2. Целевая функция (функция Растригина):

$$f(x,y) = 20 + x^2 + y^2 - 10\left(\cos 2\pi x + \cos 2\pi y\right),$$

область поиска $D = [-100:100] \times [-100;100]$. Точное решение: $x^* = y^* = 0$, $f(x^*, y^*) = 0$. Результаты решения примера отражены в табл. 3 и 4.

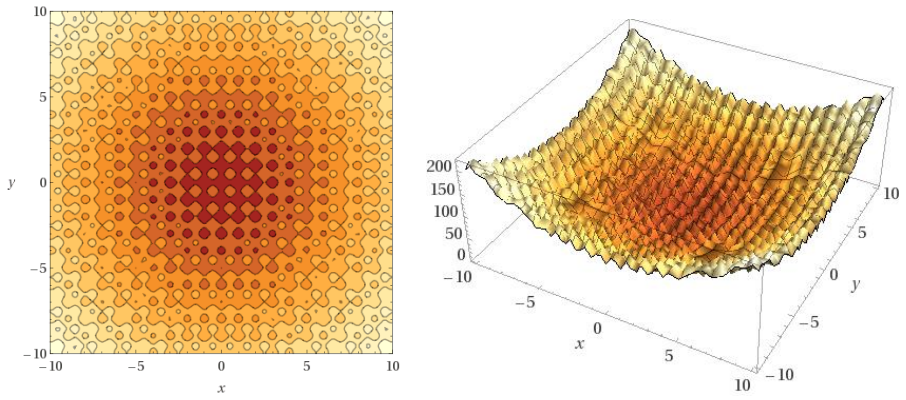

Рис.6. Линии уровня и график функции Растригина

Влияние параметров меметического алгоритма (при использовании метода муравьиных колоний). Функция Растригина

Таблица 3

Параметры меметического алгоритма					Параметры метода муравьиных колоний					$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	m	r	K	ξ	q			
25	20	10	5	0,001	20	10	10	0,85	0,0001	0,13434	0,00056	0,17866
25	20	10	5	0,1	20	10	10	0,85	0,0001	0,18627	0,00005	0,23419
50	20	10	5	0,1	10	5	20	0,85	0,0001	**0,00092**	**0**	0,00195
25	50	10	5	0,1	10	5	20	0,85	0,0001	0,00808	**0**	0,02686
25	30	20	5	0,1	10	5	20	0,85	0,0001	0,04004	**0**	0,06817

Влияние параметров меметического алгоритма (при использовании метода имитации отжига). Функция Растригина

Таблица 4

Параметры меметического алгоритма					Параметры метода имитации отжига				$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	N	T_0	C	β			
25	20	10	5	0,001	1000	1000	0,85	0,95	**0**	**0**	**0**
25	20	10	5	0,001	3000	100	0,85	0,95	**0**	**0**	**0**
25	20	10	5	0,01	3000	10	0,85	0,9	**0**	**0**	**0**
25	20	10	5	0,001	100	100	0,85	0,95	3,62271	0,375716	2,078148
25	20	10	5	0,001	500	100	0,85	0,95	8,59E-06	3,83E-09	7,24E-05
15	20	10	5	0,001	800	10	0,85	0,95	0,059713	**0**	0,237476

Пример 3. Целевая функция (функция Швефеля):

$$f(x,y) = -837{,}9658 - x \cdot \sin\sqrt{|x|} - y \cdot \sin\sqrt{|y|},$$

область поиска $D = [-100:100] \times [-100;100]$. Точное решение: $x^* = y^* = 420{,}96$, $f(x^*, y^*) = -837{,}9658$. Результаты решения примера отражены в табл. 5 и 6.

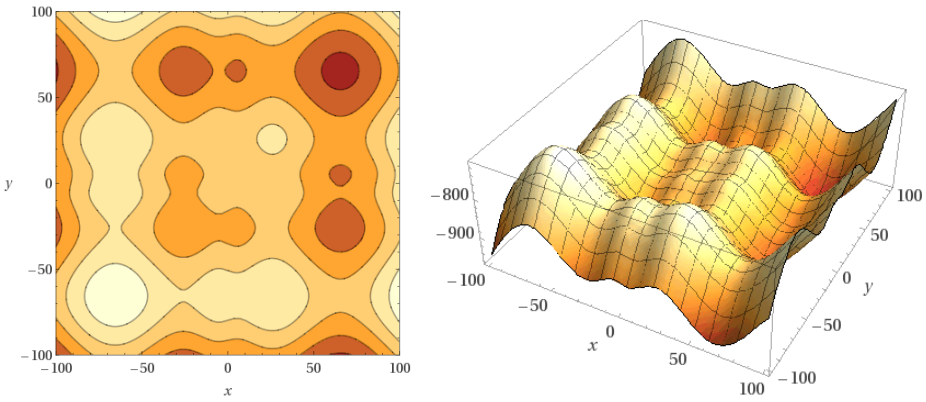

Рис.7. Линии уровня и график функции Швефеля

Влияние параметров меметического алгоритма (при использовании метода муравьиных колоний). Функция Швефеля

Таблица 5

Параметры меметического алгоритма					Параметры метода муравьиных колоний					$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	m	r	K	ξ	q			
25	20	10	5	0,001	20	10	10	0,85	0,0001	-645,82356	-829,78406	99,53866
150	40	20	10	0,1	20	10	10	0,85	0,0001	-762,13255	-837,4895	59,63854
200	20	5	3	0,1	20	10	10	0,85	0,0001	-757,67882	-837,69154	53,37359
300	20	5	3	0,001	10	5	10	0,85	0,0001	-777,00427	-832,6224	42,75873
1000	10	5	3	0,001	10	5	10	0,85	0,0001	-819,89811	-837,85295	16,14568
2500	10	5	3	0,001	10	5	10	0,85	0,0001	-832,9426	**-837,9433**	5,0068

Влияние параметров меметического алгоритма (при использовании метода имитации отжига). Функция Швефеля

Таблица 6

Параметры меметического алгоритма					Параметры метода имитации отжига				$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	N	T_0	C	β			
10	20	10	5	0,1	300	1000	0,85	0,95	-722,014	-837,284	73,239
10	20	10	5	0,1	500	1000	0,85	0,95	-719,542	-837,165	77,739
20	20	10	5	0,1	500	1000	0,85	0,95	-764,601	-836,493	51,03
50	20	10	5	0,1	100	1000	0,85	0,95	-798,928	-837,794	34,909
50	100	20	10	0,01	250	1000	0,7	0,9	-830,55	**-837,955**	7,801

Для функции Швефеля алгоритм показывает менее точные результаты, что объясняется крайне сложной и многоэкстремальной структурой функции. Для таких функций параметр σ не следует выбирать слишком большим.

Пример 4. Целевая функция (функция Букина №6)

$$f(x,y) = 100\sqrt{|\, y - 0,01x^2 \,|} + 0,01 \cdot |\, x + 10 \,|,$$

область поиска $D = [-100:100] \times [-100;100]$. Точное решение: $x^* = -10$, $y^* = 1$, $f(x^*, y^*) = 0$. Результаты решения примера отражены в табл. 7 и 8.

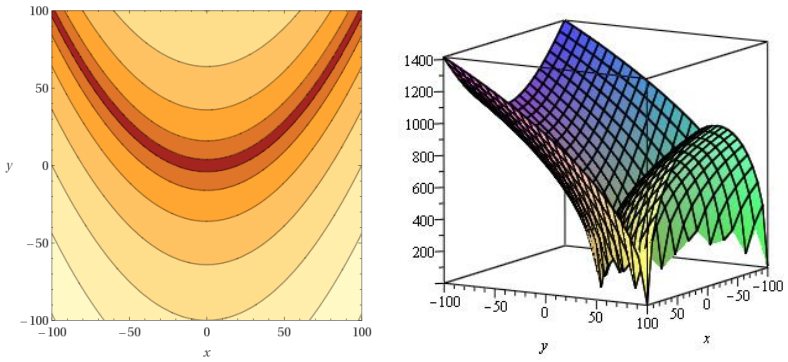

Рис.8. Линии уровня и график функции Букина №6

Влияние параметров меметического алгоритма (при использовании метода муравьиных колоний). Функция Букина №6 Таблица 7

Параметры меметического алгоритма					Параметры метода муравьиных колоний					$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	m	r	K	ξ	q			
25	20	10	5	0,001	20	10	10	0,85	0,0001	4,74811	0,36169	3,03991
25	20	10	5	0,1	20	10	10	0,85	0,0001	5,01667	0,6558	2,86372
100	10	5	3	0,1	10	5	10	0,85	0,0001	2,46723	0,25467	1,42307
200	10	5	3	0,1	10	5	10	0,85	0,0001	1,86747	0,1201	1,07266
500	10	5	33	0,01	10	5	10	0,85	0,0001	0,95661	0,12441	0,50721

Влияние параметров меметического алгоритма (при использовании метода имитации отжига). Функция Букина №6 Таблица 8

Параметры меметического алгоритма					Параметры метода имитации отжига				$\overline{f}$	$f_{наим}$	$\overline{\sigma}$
$M_{\max}$	m	K	q	σ	N	T_0	C	β			
25	20	10	5	0,01	250	1000	0,7	0,9	0,265	**0,023**	0,339
25	20	10	5	0,01	250	1000	0,7	0,8	1,022	0,016	2,436
25	20	10	5	0,01	250	1000	0,7	0,99	6,943	0,215	3,421
25	20	10	5	0,01	250	1000	0,85	0,9	0,494	0,046	1,529
50	20	10	5	0,01	200	1000	0,7	0,9	0,523	**0,025**	0,36

Данные, приведенные в табл. 1-8, показывают, что алгоритм сконструирован достаточно хорошо, и приемлемые результаты достигаются за небольшое количество итераций.

Пример 5. Задача об ориентации космического аппарата.

С помощью описанного алгоритма была решена задача об ориентации космического аппарата. Рассмотрим плоский вариант задачи ориентации космического аппарата (КА): поворот КА производится лишь в одной плоскости за счет вращения маховика, установленного внутри КА. На вал маховика подается вращательный момент M, который сообщает ему угловое ускорение $\ddot{\theta}$. Таким образом, $J_M\ddot{\theta} = M$, где J_M - момент инерции маховика. Согласно закону сохранения кинетического момента КА: $\frac{d}{dt}\left(J_M\dot{\theta} + J_0\dot{\varphi}\right) = 0$, где J_0 - момент инерции КА, φ - угол ориентации. Тогда из последнего равенства следует, что $J_M\ddot{\theta} = -J_0\ddot{\varphi} = M$. Величину, пропорциональную вращательному моменту M, взятую с обратным знаком, примем за сигнал управления $u = -\frac{M}{J_0}$.

Введем обозначения для составляющих вектора состояния: $x_1(t) = \varphi$, $x_2(t) = \dot{\varphi}$, тогда $\dot{x}_1(t) = x_2(t)$, $\dot{x}_2(t) = u(t)$. Функционал качества управления: $I = \int_{t_0}^{t_N} u^2(t)dt$.

Допустим, что в начальный момент времени $t_0 = 0$: $x_1(t_0) = x_2(t_0) = 0$, в конечный момент времени $t_N = 1$: $x_1(t_N) = \pi$, $x_2(t_N) = 0$.Тогда оптимальное управление, полученное с помощью принципа максимума: $u^*(t) = -12\pi t + 6\pi$. Точное значение критерия: $I^* = 12\pi^2 \approx 118,435$.

Для решения данной задачи используемый функционал качества управления с помощью метода дискретизации Рунге-Кутты и численного интегрирования был

сведен в функцию, зависящую от управлений в различные моменты времени функционирования системы. Моменты времени функционирования системы выступают в роли неизвестных. Полученная функция оптимизируется разработанным алгоритмом. Подробно переход от задачи оптимального управления непрерывной динамической системой к задаче поиска условного экстремума функции многих переменных изложен в [20].

На рис. 9 изображены результаты (управление и траектории), полученные меметическим алгоритмом, (сплошная линия) и точное решение задачи (штриховая линия). Полученное с помощью алгоритма значение критерия: $I = 118{,}739$. Точное значение критерия: $I^* = 12\pi^2 \approx 118{,}435$.

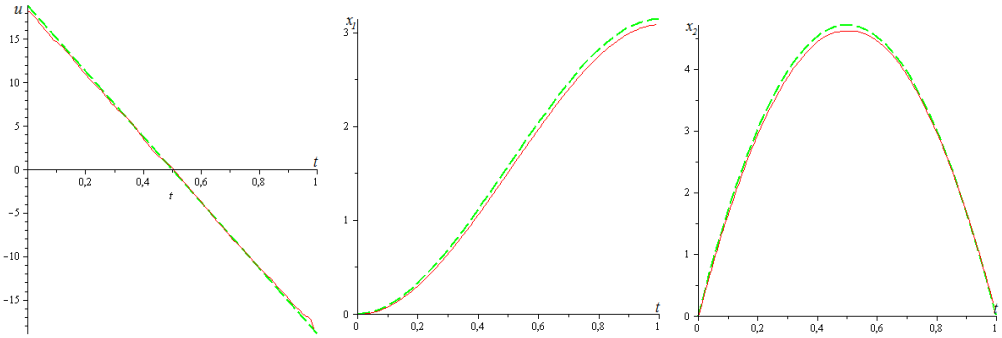

Рис. 9. Результаты работы алгоритма для задачи об ориентации космического аппарата

Полученные результаты подтверждают эффективность применения разработанного меметического алгоритма для решения задач поиска оптимального управления.

Заключение

В работе предложен меметический алгоритм для решения задач поиска условного глобального экстремума функций. Решение данного рода задач зачастую

востребовано в ходе проектирования конструкций летательных аппаратов, когда возникает необходимость оптимизации характерных параметров (вес, дальность полета, аэродинамические характеристики) и при разработке систем управления, как отдельными элементами конструкции, так и объектом в целом. В работе также реализован программный комплекс, позволяющий применять разработанный алгоритм для различных функций, а также проводить анализ эффективности его работы. Эффективность метода продемонстрирована на нескольких модельных примерах, обладающих как простой, так и сложной структурой линий уровня. Кроме того, описано применение данного алгоритма для решения задачи об ориентации космического аппарата.

Библиографический список

1. Попов В.И. Системы ориентации и стабилизации космических аппаратов – М.: Машиностроение, 1986. – 184 с.

2. Раушенбах Б.В., Токарь Е.Н. Управление ориентацией космических аппаратов – М.: Наука, 1974. — 600 с.

3. Крылов И.А. Численное решение задачи об оптимальной стабилизации спутника // Журнал вычислительной математики и математической физики: 1968. Т.8, №1, С.203-208.

4. Dawkins R. Universal Darwinism in D.S. Bendall (ed.), Evolution: From Molecules to Men, Cambridge University Press, Cambridge, 1983, P. 403-425.

5. Dawkins R. The Selfish Gene. Oxford University Press, 1976, 192 p.

6. Moscato P. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms // Caltech Concurrent Computation Program (report 826), 1989.

7. Krasnogor N. Coevolution of genes and memes in memetic algorithms. Graduate Student Workshop. 371 p.

8. Kendall G., Soubeiga E., Cowling P. Choice function and random hyperheuristics // 4th Asia-Pacific Conference on Simulated Evolution and Learning, SEAL 2002, P. 667–671.

9. Ong Y.S., Keane A.J. Meta-Lamarckian learning in memetic algorithms // IEEE Transactions on Evolutionary Computation. V.8 (2), 2004. P. 99–110.

10. Ong Y.S., Lim M.H., Zhu N., Wong K.W. Classification of Adaptive Memetic Algorithms: A Comparative Study // IEEE Transactions on Systems Man and Cybernetics, Part B. 36 (1): 141.

11. Smith J.E. Coevolving Memetic Algorithms: A Review and Progress Report // IEEE Transactions on Systems Man and Cybernetics - Part B 37 (1), 2007. P. 6-17.

12. Krasnogor N., Gustafson S. Toward truly "memetic" memetic algorithms: discussion and proof of concepts, Advances in Nature-Inspired Computation: the PPSN VII Workshops. PEDAL (Parallel Emergent and Distributed Architectures Lab), University of Reading, 2002.

13. Ichimura T., Kuriyama Y. Learning of neural networks with parallel hybrid GA using a royal road function // IEEE International Joint Conference on Neural Networks. 2. New York, NY, 1998. pp 1131–1136.

14. Aguilar J., Colmenares A. Resolution of pattern recognition problems using a hybrid genetic/random neural network learning algorithm// Pattern Analysis and Applications. V. 1 (1), 1998. pp. 52-61.

15. Wehrens R., Lucasius C., Buydens L., Kateman G. HIPS, A hybrid self-adapting expert system for nuclear magnetic resonance spectrum interpretation using genetic algorithms// Analytica Chimica ACTA. V. 277 (2), 1993. pp. 313-324.

16. Dorigo M., Socha K. Ant colony optimization for continuous domains // European Journal of Operational Research. V. 185, 2008. pp. 1155-1173.

17. Пантелеев А.В. Метаэвристические алгоритмы поиска глобального экстремума. - М.: Изд-во МАИ-ПРИНТ, 2009. - 129 с.

18. Пантелеев А.В., Алешина Е.А. Применение непрерывной модификации метода муравьиных колоний к задаче поиска оптимального управления дискретными детерминированными системами // Научный вестник МГТУ ГА. 2013. Вып. 194(8). С. 47- 54.

19. Пантелеев А.В., Дмитраков И.Ф. Анализ сравнительной эффективности метода имитации отжига для поиска глобального экстремума функций многих переменных // Научный Вестник МГТУ ГА.2009. Вып. 145(8). С. 26-31.

20. Пантелеев А.В., Письменная В.А. Применение меметического алгоритма в задаче поиска оптимального программного управления нелинейными непрерывными детерминированными системами // Авиакосмическое приборостроение – М.: 2014. №3. С. 26-34.