УДК 004.8

А. А. Ковалевич, аспирант,
А. И. Якимов, канд. техн. наук, доц.,
Д. М. Албкеират, аспирант,
ГУВПО "Белорусско-Российский университет",
г. Могилев, Беларусь,
e-mail: ykm@tut.by

Исследование стохастических алгоритмов оптимизации для применения в имитационном моделировании систем

Проведен сравнительный анализ стохастических алгоритмов оптимизации: роя частиц, имитации отжига и дифференциальной эволюции. Для проверки алгоритмов использовались функции Branin RCOS, модифицированная версия RCOS, Easom и Goldstein-Price. Показано, что алгоритм дифференциальной эволюции является лучшим алгоритмом, так как стабильно находит оптимум функции за минимальное время.

Ключевые слова: *стохастический алгоритм оптимизации, рой частиц, имитация отжига, дифференциальная эволюция, имитационное моделирование*

Введение

Основной идеей имитационного моделирования часто является построение модели для последующей оптимизации. Поэтому этап оптимизации имитационных моделей является ключевым для всего имитационного моделирования [1]. От эффективности оптимизации модели напрямую зависит эффективность моделирования для конечного пользователя модели. В настоящее время существует множество алгоритмов поиска оптимального решения, поэтому выбор действительно лучшего из них является довольно трудной задачей. Для выбора эффективного алгоритма используются критерии оптимальности найденного решения (то есть, какую часть от заранее известного оптимального решения составляет решение, найденное с помощью выбранного алгоритма) и времени, затраченного на поиск этого решения.

Анализ пакетов оптимизации, используемых при имитационном моделировании, показывает, что в имитационном эксперименте используются алгоритмы случайного поиска. Наиболее известные оптимизационные пакеты, применяемые в имитационном моделировании, следующие: *AutoStat* (AutoSimulations, Inc.; www.autosim.com) — использует эволюционные стратегии, генетические алгоритмы, применяется в системе имитации AutoMod; *Evolutionary Optimizer* (ImagineThat, Inc; www.imaginethatinc.com) — основан на эволюционных стратегиях, генетических алгоритмах, применяется в Extend; *OptQuest* (OptTek Systems, Inc.; www.opttek.com) — использует случайный поиск, поиск с запретами, нейронные сети, встроен в системы имитации AnyLogic, Arena, Crystal Ball, CSIM19, Enterprise Dynamics, Micro Saint, ProModel, Quest, SimFlex, SIM-PROCESS, SIMUL8, TERAS; пакет *RISKOptimizer* (Palisade Corp.; www.palisade.com), использующий генетические алгоритмы, применяется в @RISK; *WITNESS Optimizer* (banner Group, Inc.; www.lanner.com/corporate) — использует алгоритм имитации отжига, поиск с запретами, применяется в системе имитационного моделирования WITNESS [2, 3].

Преимущества эволюционных подходов к решению оптимизационных задач: повышенное быстродействие; высокая надежность и помехоустойчивость; высокая робастность, т. е. малая чувствительность к нерегулярностям поведения целевой функции; сравнительно простая внутренняя реализация; малая чувствительность к росту размерности задачи оптимизации; возможность естественного ввода в процесс поиска операции обучения и самообучения; легкое построение новых алгоритмов, реализующих различные эвристические процедуры адаптации в рамках известных схем случайного поиска.

Недостатки их использования: отсутствие гарантий качества получаемых решений; невозможность проверки на всех мыслимых классах задач; необходимость предварительной настройки параметров алгоритма для его эффективной работы.

В данной статье проводится сравнение трех алгоритмов стохастической оптимизации: алгоритма имитации отжига (ИО), роя частиц (РЧ) и дифференциальной эволюции (ДЭ).

1. Алгоритм имитации отжига

Алгоритм имитации отжига — общий алгоритмический метод решения задачи глобальной оптимизации, особенно дискретной и комбинаторной оптимизации, является одним из примеров метода Монте-Карло и относится к классу пороговых алгоритмов. Он создан в 1983 году [4]. Алгоритм основан на имитации процесса кристаллизации вещества при переходе из жидкого состояния в твердое. Процесс протекает при постоянно понижающейся температуре. При этом предполагается, что атомы уже выстроились в кристаллическую решетку, но допустимы переходы отдельных атомов из одной ячейки в другую. Переход происходит с некоторой вероятностью, которая

уменьшается с понижением температуры. Устойчивая кристаллическая решетка соответствует минимуму энергии атомов, поэтому атом переходит в состояние с меньшим уровнем энергии либо остается на месте.

Пусть $f_{\min}$: $R^D \to R^+$ — целевая функция, которую требуется минимизировать. Схема работы алгоритма ИО, используемого для проведения исследований, выглядит следующим образом.

Шаг 1. Положить $i = 0$, $i \in I = \{0, 1, 2, ...\}$. Выбрать начальное решение $\mathbf{x}_i = x_1, x_2, ..., x_D$ из пространства решений X и начальную температуру T_i. При этом скорость и закон убывания температуры задаются, например, по формуле

$$T_i = 1 \cdot 10^{19} \cdot 0{,}95^i | i \in I. \quad (1)$$

Шаг 2. Оценить начальное решение по известной целевой функции $f(\mathbf{x}_i)$.

Шаг 3. Случайным образом изменить решение $\mathbf{x}_i$ и получить $\mathbf{x}^*$.

Шаг 4. Оценить полученное решение $f(\mathbf{x}^*)$.

Шаг 5. Проверить возможную замену текущего решения $\mathbf{x}_i$ измененным решением $\mathbf{x}^*$. Точка $\mathbf{x}^*$ становится точкой x_{i+1} с вероятностью $P(\mathbf{x}_{i+1} := \mathbf{x}^* | \mathbf{x}_i)$, которая вычисляется в соответствии с распределением Гиббса:

$$P(\mathbf{x}_{i+1} := \mathbf{x}^* | \mathbf{x}_i) = \begin{cases} 1 & | f(\mathbf{x}^*) - f(\mathbf{x}_i) < 0; \\ \exp\left(-\dfrac{f(\mathbf{x}^*) - f(\mathbf{x}_i)}{T_i}\right) & | f(\mathbf{x}^*) - f(\mathbf{x}_i) \geqslant 0, \end{cases}$$

где T_i — элементы произвольной убывающей, сходящейся к нулю положительной последовательности, являющейся аналогом падающей температуры в кристалле (см. (1) на шаге 1).

Шаг 6. Уменьшить температуру, т. е. задать $i := i + 1$. При этом температура уменьшится в соответствии с формулой (1).

Шаг 7. Шаги 3—6 повторять, пока не выполнен критерий останова $T_i < 1 \cdot 10^{-322}$, принятый в настоящих исследованиях.

Алгоритм ИО достаточно широко распространен благодаря своей простоте, гибкости и эффективности, поскольку для данного алгоритма удается аналитически исследовать его свойства и доказать асимптотическую сходимость [4].

2. Алгоритм роя частиц

Алгоритм роя частиц — метод численной оптимизации, для использования которого не требуется знать точного градиента оптимизируемой функции. Алгоритм РЧ был доказан Кеннеди, Эберхартом и Ши и изначально предназначался для имитации социального поведения. Алгоритм был упрощен и было замечено, что он пригоден для выполнения оптимизации. Алгоритм РЧ оптимизирует функцию, поддерживая популяцию возможных решений, называемых частицами, и перемещая эти частицы в пространстве решений согласно простой формуле. Перемещения подчиняются принципу наилучшего найденного в этом пространстве положения, которое постоянно изменяется при нахождении частицами более выгодных положений [5].

Пусть $f_{\min}$: $R^D \to R^+$ — целевая функция, выбраны параметры этой функции $\mathbf{x} = (x_1, ..., x_D)$, которые необходимо оптимизировать, и определена область для поиска оптимальных значений: $\mathbf{b}_{\sup}$, $\mathbf{b}_{\inf}$ — верхняя и нижняя границы пространства решений соответственно; $|S|$ — число частиц в рое S (в данной работе $|S| = 60$), каждой из которых сопоставлена координата $\mathbf{x}_i \in R^D$ в пространстве решений и скорость $\mathbf{v}_i \in R^D$. Пусть также $\mathbf{p}_i$ — лучшее из известных положений i-й частицы (ЛПЧ); $\mathbf{g}$ — наилучшее известное положение роя (НПР) в целом. Схема работы алгоритма РЧ выглядит следующим образом.

Шаг 1. Сгенерировать начальное положение каждой частицы с помощью случайного вектора

$$\mathbf{x}_i := \text{rand}(\mathbf{b}_{\sup}, \mathbf{b}_{\inf}) \mid i \in S = \{1, 2, ..., |S|\},$$

имеющего многомерное равномерное распределение. Задаются скорости их движения

$$\mathbf{v}_i := \text{rand}(-(\mathbf{b}_{\sup} - \mathbf{b}_{\inf}), (\mathbf{b}_{\sup} - \mathbf{b}_{\inf})).$$

Шаг 2. Инициализация $\mathbf{p}_i$ и $\mathbf{g}$. Для каждой частицы рассчитывается функция пригодности $f(\mathbf{x}_i)$. Присвоить лучшему известному положению частицы его начальное положение $\mathbf{p}_i := \mathbf{x}_i$. Лучшая частица с точки зрения целевой функции f объявляется значением НПР:

$$\mathbf{g} := \mathbf{p}_i | f(\mathbf{p}_i) \to \min.$$

Шаг 3. Сгенерировать случайные векторы

$$\mathbf{r}_p := \text{rand}(-1, +1) \text{ и } \mathbf{r}_g := \text{rand}(-1, +1).$$

Введение $\mathbf{r}_p$ и $\mathbf{r}_g$ в оптимизацию предназначено для моделирования незначительного непредсказуемого реального поведения роя.

Шаг 4. Корректировка скорости частицы. Скорость частицы меняется в соответствии с взаимным расположением позиций ЛПЧ и НПР. Они стремятся в направлении этих позиций в соответствии с формулой

$$\mathbf{v}_i := \omega \mathbf{v}_i + \varphi_p \mathbf{r}_p \times (\mathbf{p}_i - \mathbf{x}_i) + \varphi_g \mathbf{r}_g \times (\mathbf{g} - \mathbf{x}_i),$$

где операция $\times$ означает покомпонентное умножение.

Шаг 5. Обновить положение частицы переносом $\mathbf{x}_i$ на вектор скорости

$$\mathbf{x}_i := \mathbf{x}_i + \mathbf{v}_i.$$

Частица может вылетать за пределы разрешенной области, но полученные ею значения не учитываются до тех пор, пока она не вернется обратно.

Шаг 6. Если $f(\mathbf{x}_i) < f(\mathbf{p}_i)$, то обновить ЛПЧ $\mathbf{p}_i := \mathbf{x}_i$. Если $f(\mathbf{p}_i) < f(\mathbf{g})$, то обновить НПР $\mathbf{g} := \mathbf{p}_i$. Теперь $\mathbf{g}$ содержит лучшее из найденных решений.

Шаг 7. Шаги 3—6 выполняются для каждой частицы, пока не выполнен критерий останова (например, достижение заданного числа итераций или необходимого значения целевой функции). В исследованиях критерием останова является число итераций, равное $1 \cdot 10^4$.

Алгоритм имеет три параметра: ω, φ_p и φ_g. При проведении экспериментов использовали следующие значения: $\omega = 0{,}93$, $\varphi_p = 0{,}7$ и $\varphi_g = 1{,}2$.

Алгоритм РЧ можно эффективно распределить на несколько параллельных процессов, за счет чего повысится его быстродействие. По сравнению с генетическим алгоритмом, операторы которого могут быть реализованы различным образом, имеется лишь один оператор — вычисление скорости, что делает его более простым в использовании.

3. Алгоритм дифференциальной эволюции

Дифференциальная эволюция — алгоритм многомерной математической оптимизации, относящийся к классу стохастических алгоритмов оптимизации (т. е. работает с использованием случайных чисел) и использующий некоторые идеи генетических алгоритмов.

Алгоритм ДЭ предназначен для нахождения глобального экстремума недифференцируемых, нелинейных, мультимодальных (имеющих, возможно, большое число локальных экстремумов) функций от многих переменных. Алгоритм был предложен Р. Сторном и К. Прайсом, впервые опубликован ими в 1995 г. и разработан в дальнейшем в их более поздних работах. На первом международном конкурсе по эволюционным вычислениям, который проводился в мае 1996 г., алгоритм ДЭ занял третье место, при этом он был лучшим среди генетических алгоритмов и проиграл только двум алгоритмам, которые не являются в общем случае универсальными, однако быстрее решили предложенный набор задач.

Подобно другим эволюционным алгоритмам ДЭ рассматривает случайную популяцию решений. Пусть начальная популяция решений состоит из NP индивидов и пространство поиска является D-мерным. Популяция n-й итерации может быть представлена следующим образом:

$$\mathbf{X}(n) = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{NP}),$$

где $\mathbf{x}$ — возможное решение в D-мерном пространстве поиска. Пусть существует некоторый критерий качества $f(\mathbf{x})$ такой, чтобы найти

$$\mathbf{x}^* : f(\mathbf{x}^*) = \min_{\mathbf{x}} f(\mathrm{x}).$$

Пусть f: $R^D \to R^+$ — целевая функция, которую требуется минимизировать. Алгоритм ДЭ включает три эволюционных процесса: мутацию, скрещивание, выбор.

Шаг 1. Оператор мутации случайным образом выбирает три различных индивида из текущей популяции и создает нового измененного индивида

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \times (\mathbf{x}_{r2} - \mathbf{x}_{r3}) | F \in [0, 2],$$

где F — управляющий параметр, обычно выбираемый в интервале [0, 2], с наилучшими значениями в диапазоне [0,5, 0,9]. Оператор мутации является стохастическим отображением

$$T_m : S^{NP} \to S,$$

где $S = R^D$; S^{NP} — пространство популяции (в исследованиях $NP = 60$).

Шаг 2. Оператор скрещивания создает новое решение копированием компонентов мутационного вектора $\mathbf{v}_i$ и выбранного вектора $\mathbf{x}_i$:

$$u_{ji} = \begin{cases} v_{ji} | r_b \leqslant CR \wedge j = r_r, \; i = 1, ..., NP; \\ x_{ji} | r_b > CR \vee j \neq r_r, \; i = 1, ..., D, \end{cases}$$

где $r_b = \mathrm{rand}[0, 1]$ — случайное число в интервале [0, 1]; $r_r =]\mathrm{rand}[0, D][$ — случайное целое число в интервале [1, D]; $CR \in [0, 1]$ — управляющий параметр скрещивания.

Оператор скрещивания является стохастическим отображением

$$T_r : S^2 \to S.$$

Шаг 3. Оператор выбора является детерминированным процессом в алгоритме ДЭ и выбирает индивида с лучшим значением целевой функции для следующего поколения:

$$\mathbf{x}_i(n + 1) = \begin{cases} \mathbf{u}_i \mid f(\mathbf{u}_i) \leqslant f(\mathbf{x}_i); \\ \mathbf{x}_i \mid f(\mathbf{u}_i) > f(\mathbf{x}_i). \end{cases}$$

Процесс выбора описывается детерминированным оператором

$$T_s : S^2 \to S.$$

Оператор выбора гарантирует, что лучшее значение целевой функции не может быть пропущено, что приводит к быстрой сходимости. Алгоритм ДЭ может быть описан следующим образом:

$$\mathbf{X}(n + 1) = \{\mathbf{x}_i(n + 1) | \mathbf{x}_i(n + 1) = \\ = T_s \circ T_r \circ T_m(\mathbf{X}(n)), \; i = 1, ..., NP\}.$$

После этого шаги 1—3 повторяются, пока не будет сформировано нужное число поколений, которое в исследованиях принято равным $1 \cdot 10^4$.

Алгоритм прост в реализации и использовании (содержит малое число управляющих параметров, требующих подбора: коэффициент мутации F и вероятность скрещивания CR), легко распараллеливается [6, 7].

4. Функции, используемые для проверки алгоритмов

Для проверки алгоритмов использовали функции *Branin RCOS*, модифицированная версия *RCOS*, *Easom* и *Goldstein-Price*. Этот выбор обусловлен тем фактом, что функции специально предложены и используются для тестирования алгоритмов оптимизации, основанных на случайном поиске.

Для функции *Branin RCOS* характерно наличие множества локальных оптимумов. Определение функции *Branin RCOS*:

$$Branin(x_1, x_2) =$$

$$= (x_2 - b(x_1)^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e, \quad (2)$$

где $b = 5{,}1/4\pi^2$; $c = 5/\pi$; $d = 6$; $e = 10$; $f = 1/8\pi$; $-5 \leqslant x_1 \leqslant 10$; $0 \leqslant x_2 \leqslant 15$.

Глобальный минимум этой функции:

$$Branin(x_1, x_2) = 0{,}397887 | (x_1, x_2) =$$

$$= (9{,}42478;\ 2{,}475),\ (-\pi;\ 12{,}275),\ (\pi;\ 2{,}275). \quad (3)$$

График функции (2) представлен на рис. 1.

Функция *Easom* характеризуется наличием пологих участков и глобального минимума, площадь которого достаточно мала по сравнению с площадью пространства поиска. Определяется функция *Easom* следующим образом:

$$Easom(x_1, x_2) =$$

$$= -\cos(x_1)\cos(x_2)\exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2)), \quad (4)$$

где $-100 \leqslant x_i \leqslant 100$, $i = 1, 2$.

Глобальный минимум этой функции:

$$\text{Easom}(x_1, x_2) = -1{,}0 \mid (x_1, x_2) = (\pi, \pi). \quad (5)$$

График функции (4) представлен на рис. 2.

Для модифицированной функции *RCOS* характерно наличие шести локальных экстремумов и одного глобального. Определение функции:

$$M\_Branin(x_1, x_2) = 1/[a(x_2 + bx_1^2 + cx_1 - d)^2 +$$

$$+ e(1 - f)\cos(x_1)\cos(x_2) + \log(x_1^2 + x_2^2 + 1) + e], \quad (6)$$

где $a = 1$, $b = 5{,}1/4\pi$, $c = 5/\pi$, $d = 6$, $e = 10$, $f = 1/8\pi$.

Область поиска: $-5 \leqslant x_1 \leqslant 10$, $0 \leqslant x_2 \leqslant 15$.

Глобальный максимум этой функции:

$$M\_Branin(x_1, x_2) = 0{,}689087 \mid (x_1, x_2) =$$

$$= (3{,}06699;\ 2{,}99805 \cdot 10^{-16}). \quad (7)$$

График функции (6) представлен на рис. 3.

Для функции *Goldstein-Price* характерно наличие нескольких оптимумов, разделенных широким плато, один из которых — глобальный мак-

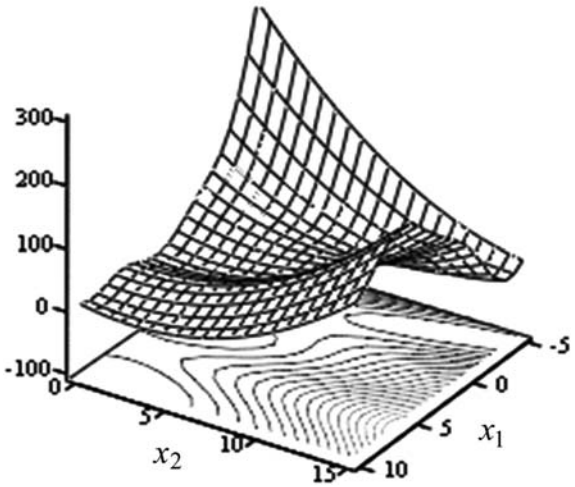

Рис. 1. График функции *Branin*(x_1, x_2)

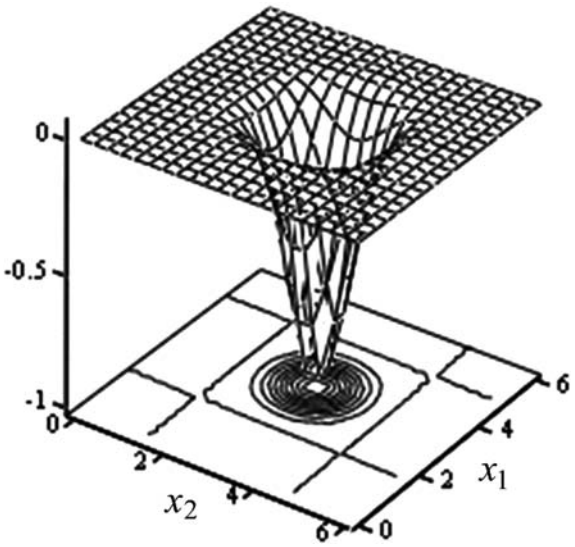

Рис. 2. График функции *Easom*(x_1, x_2)

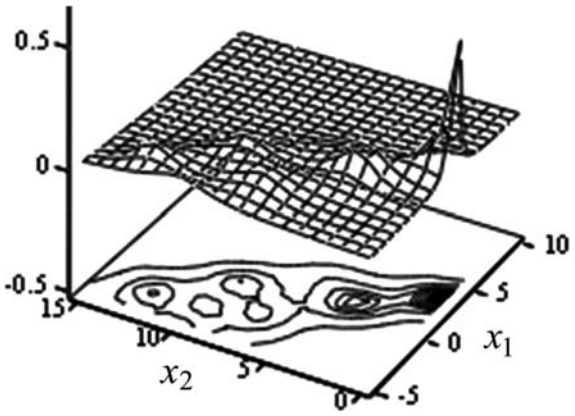

Рис. 3. График функции *M_Branin*(x_1, x_2)

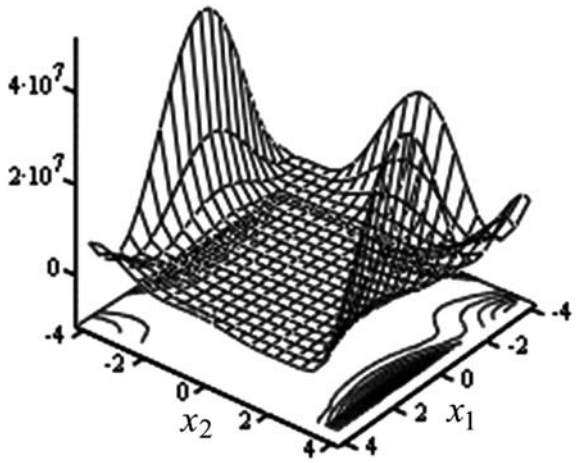

Рис. 4. График функции *Golden*(x_1, x_2)

симум. Определяется функция *Goldstein-Price* выражением

$$Gold(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)], \quad (8)$$

где $-2 \leqslant x_i \leqslant 2$, $i = 1, 2$.

Глобальный минимум этой функции

$$Gold(x_1, x_2) = 3{,}0 \mid (x_1, x_2) = (0; -1). \quad (9)$$

График функции (8) представлен на рис. 4.

5. Результаты исследований и их обсуждение

Для проведения исследований использовали компьютер с процессором *AMD Athlon* 64 3500+ и 1 ГБ ОЗУ под управлением операционной системы *Windows XP SP3*. Реализации алгоритмов были написаны на языке C++.

При анализе чувствительности параметров F и CR алгоритма ДЭ было отмечено, что алгоритм не чувствителен к этим параметрам для всех функций кроме модифицированной функции *RCOS*. Результаты исследования чувствительности параметров F и CR для модифицированной функции *RCOS* приведены в табл. 1.

Для проведения исследований использовали следующие значения параметров алгоритма ДЭ: $F = 0{,}9$; $CR = 0{,}5$.

При исследовании функции *Branin RCOS* глобальный минимум был найден только с помощью алгоритма ДЭ (табл. 2). Так как с помощью ДЭ на поиск лучшего решения было потрачено не больше времени, то следует признать, что для функции *Branin RCOS* оптимальным алгоритмом поиска глобального минимума является алгоритм ДЭ. Далее следует алгоритм роя частиц, который находил в среднем решение за то же время, но на 0,04 % хуже. Последним оказался алгоритм имитации отжига, с помощью которого решения находились медленнее на 3,8 % и на 10,5 % хуже оптимального.

При исследовании функции *Easom* (табл. 3) глобальный минимум был найден только с помощью алгоритма ДЭ. В среднем алгоритм роя частиц находил решение на 0,21 %, а имитации отжига — на 13,04 % хуже. Учитывая, что с помощью ДЭ на поиск лучшего решения было потрачено еще и меньшее время (на 22,6 % и на 13,1 % быстрее, чем с помощью алгоритмов роя частиц и имитации отжига соответственно), следует считать, что для функции *Easom* оптимальным алгоритмом поиска глобального минимума является алгоритм ДЭ. Далее следует алгоритм РЧ и худшее решение получено с помощью алгоритма ИО.

Таблица 1

Результаты исследований чувствительности параметров ДЭ для функции *M_Branin* (x_1, x_2)

Коэффициент мутации F	Вероятность скрещивания CR	Решение
0,1	0,1	0,167853
0,1	0,5	0,628472
0,1	0,9	0,167826
0,9	0,1	0,167853
0,9	0,9	0,689087
2	0,1	0,689087
2	0,5	0,689087
2	0,9	0,689087

Таблица 2

Результаты исследований для функции *Branin* (x_1, x_2)

Наименование алгоритма	Время работы, с	Решение	Оптимальность решения, %
Дифференциальная эволюция	0,556	0,397887	100
Рой частиц	0,556	0,398056	99,96
Имитация отжига	0,577	0,439660	89,50

Таблица 3

Результаты исследований для функции *Easom* (x_1, x_2)

Наименование алгоритма	Время работы, с	Решение	Оптимальность решения, %
Дифференциальная эволюция	0,718	−1,0	100
Рой частиц	0,880	−0,997890	99,79
Имитация отжига	0,812	−0,869631	86,96

Таблица 4

Результаты исследований для функции *M_Branin* (x_1, x_2)

Наименование алгоритма	Время работы, с	Решение	Оптимальность решения, %
Дифференциальная эволюция	0,609	0,68909	100
Рой частиц	0,703	0,68845	99,91
Имитация отжига	0,765	0,68831	99,89

Таблица 5

Результаты исследований для функции *Golden* (x_1, x_2)

Наименование алгоритма	Время работы, с	Решение	Оптимальность решения, %
Дифференциальная эволюция	0,562	3,0	100
Рой частиц	0,666	3,00295	99,90
Имитация отжига	0,723	3,00667	99,78

При исследовании модифицированной функции *RCOS* (табл. 4) глобальный максимум был найден только с помощью алгоритма ДЭ. Учитывая, что с помощью ДЭ на поиск лучшего решения было потрачено еще и меньшее время, следует считать, что для модифицированной функции *RCOS* оптимальным алгоритмом поиска глобального максимума является алгоритм ДЭ. Далее следует алгоритм роя частиц. Стоит отметить, что для данной функции алгоритм имитации отжига нашел решение, которое хуже оптимального на 0,11 %, алгоритм роя частиц нашел решение на 0,09 % хуже оптимального.

При исследовании функции *Goldstein-Price* (табл. 5) алгоритм РЧ в среднем находил решение на 0,1 % хуже и на 18,5 % медленнее, чем алгоритм ДЭ. Алгоритм ИО отставал от алгоритма ДЭ на 0,22 % по оптимальности решения и на 28,6 % по скорости. Поэтому оптимальным алгоритмом следует считать алгоритм ДЭ [8].

Таким образом, для четырех функций (*Branin RCOS*, модифицированной *RCOS*, *Goldstein-Price* и *Easom*) наиболее быстрым и точным алгоритмом из трех исследуемых алгоритмов является алгоритм дифференциальной эволюции. За ним следует алгоритм роя частиц. Замыкает тройку алгоритм имитации отжига. Это можно объяснить тем фактом, что алгоритм ИО лучше всего подходит для оптимизации дискретных функций, а для функций, которые не являются дискретными, оптимальное решение, найденное с его помощью, как правило, не является глобальным.

Также стоит заметить, что при проведении экспериментов над всеми функциями алгоритм ДЭ всегда выдавал одно и то же оптимальное решение для каждой исследуемой функции, в то время как алгоритмы РЧ и алгоритм ИО каждый раз выдавали разное оптимальное решение. Таким образом, для выбранных функций алгоритм дифференциальной эволюции характеризуется стабильностью поиска решений, в отличие от двух других алгоритмов.

Заключение

Алгоритм дифференциальной эволюции является лучшим алгоритмом, так как стабильно находит оптимум функции и за минимальное время. Учитывая еще и исследования [9], где алгоритм дифференциальной эволюции сравнивался с генетическим алгоритмом, метагенетическим алгоритмом, методом "ветвей и границ" и также оказался лучшим, то следует признать, что алгоритм дифференциальной эволюции является лидирующим алгоритмом для оптимизации непрерывной функции от многих переменных. Еще одной положительной особенностью алгоритма является то, что его можно использовать и в режиме распределенных вычислений для сокращения времени проведения этапа оптимизации.

Работа выполнена при финансовой поддержке гранта Ф09М-171 БРФФИ.

Список литературы

1. **Якимов А. И.** Технология имитационного моделирования систем управления промышленных предприятий: монография. Могилев: Белорус. Рос. ун-т, 2010. — 304 с.

2. **Law A. M., McComsas M. G.** Simulation-Based Optimization // Proc. of 2002 Winter Simulation Conference. San Diego: IEEE Press, 2002. P. 41—44.

3. **Fu M. C., Glover F., April J.** Simulation Optimization: A Review, New Developments, and Applications // Proc. of 2005 Winter Simulation Conference. Orlando: IEEE Press, 2005. P. 83—95.

4. **Kirkpatrick S., Gelatt C. D., Vecchi M. P.** Optimization by simulated annealing // Science. — 1983. — № 4598. — P. 671—680.

5. **Weise T.** Global Optimization Algorithms. Theory and Application 2nd Ed. 2008. 652 p. URL: http://www.it-weise.de/projects/book.pdf. — Дата доступа 20.06.2010.

6. **Price K. V., Storn R. M., Lampinen J. A.** Differential evolution. A practical approach to global optimization. Leipzig: Springer-Verlag Berlin Heidelberg, 2005. 539 p.

7. **Zhang X. Y., Duan H. B., Yu Y. X.** Receding horizon control for multi-UAVs close formation control based on differential evolution // Science China. Information Sciences. — 2010. — Vol. 53. — № 2. — P. 223—235.

8. **Ковалевич А. А.** Сравнительный анализ алгоритмов многопараметрической оптимизации в имитационном моделировании систем // Математическое и имитационное моделирование систем. МОДС 2010: тез. докл. пятой науч.-практич. конф. с междунар. участием, 21—25 июня 2010 г. Киев: ИПММС НАН Украины, 2010. С. 218—220.

9. **Пупков К. А., Феоктистов В. А.** Алгоритм дифференциальной эволюции для задач технического проектирования // Информационные технологии. 2004. № 8. С. 25—31.