УДК 681.3.05 **DOI 10.52928/2070-1624-2022-38-4-26-31**

АЛГОРИТМ И ПРОГРАММА ЧИСЛЕННОЙ ОПТИМИЗАЦИИ, РЕАЛИЗУЮЩИЕ МЕТОД РОЯ ЧАСТИЦ

канд. техн. наук, доц. А. Ф. ОСЬКИН
(Полоцкий государственный университет);
Д. А. ОСЬКИН
(Белорусский государственный экономический университет, Минск)

Рассматривается один из методов численной оптимизации, реализующий так называемый метод роя частиц. Предлагается модификация метода, основанная на разбиении итерационного процесса вычисления на два этапа. Для ускорения вычислений и снижения их сложности. на первом этапе целевая функция заменяется упрощенной моделью, что позволяет быстро определить примерную область локализации экстремума. Окончательное решение ищется в найденной области локализации, с использованием исходной целевой функции. Описывается консольное приложение, реализующее алгоритм и приводятся результаты численных экспериментов, выполненных с помощью данного приложения.

Ключевые слова: *численная оптимизация, метод роя частиц, программная реализация метода роя частиц.*

Введение. Метод роя частиц (Particle Swarm Optimization – PSO) – один из многочисленной группы методов искусственного интеллекта, получившей название «Методы роевого интеллекта». Он применяется для численного решения оптимизационных задач и наиболее эффективен в случае сложных целевых функций.

Впервые идея метода была изложена в статье Дж. Кеннеди (J. Cennedy) и Р. Эберхарта (R. Eberhart) [1].

Исходно метод разрабатывался как инструмент для имитационного моделирования социального поведения, так как один из его авторов, Джеймс Кеннеди, – известный социальный психолог. Однако затем было замечено, что метод может быть эффективно использован для решения задач оптимизации. Алгоритм был упрощен и дальнейшее его развитие пошло именно в этом направлении.

По итогам своих исследований Дж. Кеннеди и Р. Эберхарт выпустили книгу «Swarm Intelligence» [2] («Роевой интеллект»), в которой описываются многие философские аспекты метода роя частиц и так называемого «роевого интеллекта».

Серьезный вклад в развитие метода роя частиц внес Р. Поли [3; 4], который показал, что этот метод является наиболее простым методом эволюционного программирования, обеспечивающим наиболее высокую скорость сходимости в случае удачного подбора управляющих параметров.

Метод роя частиц – итеративный процесс. Его работа в классической версии алгоритма может быть описана следующей системой уравнений[1] [5]:

$$\boldsymbol{v}_i(t+1) = w \cdot \boldsymbol{v}_i(t) + [c_1 \cdot r_1 \cdot (\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)] + [c_2 \cdot r_2 \cdot (\boldsymbol{g}(t) - \boldsymbol{x}_i(t)]$$
$$\boldsymbol{x}_i\,(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1)$$

где $\boldsymbol{v}_i\left(t+1\right)$ – скорость *i*-й частицы роя в момент $t+1$. Полужирное выделение здесь означает, что скорость частицы – это вектор, а не просто скалярное значение;

w – константа, моделирующая инерцию в движении частицы;

$\boldsymbol{v}_i(t)$ – текущая скорость *i*-й частицы;

c_1, r_1 – константы, задаваемые модельером: c_1 – это так называемая персональная или локальная весовая доля, отображающая вклад данной частицы в общий процесс моделирования, r_1 – случайная переменная в диапазоне [0, 1);

$\boldsymbol{p}_i(t)$ – лучшая позиция *i*-й частицы на данный момент;

$\boldsymbol{x}_i(t)$ – текущая позиция *i*-й частицы;

c_2, r_2 – константы, задаваемые модельером: c_2 – социальная или глобальная весовая доля, r_2 – случайная переменная в диапазоне [0, 1];

[1] Искусственный интеллект. Метод роя частиц. [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/archive/msdn-magazine/2011/august/artificial-intelligence-particle-swarm-optimization.

$\boldsymbol{g}(t)$ – лучшая позиция на данный момент для любой из частиц роя;

$\boldsymbol{x}_i(t+1)$ – новая позиция *i*-й частицы роя.

Как показано в [1–3], такой процесс сходится и приводит к нахождению экстремума оптимизируемой функции.

Основным достоинством метода роя частиц является отсутствие каких-либо особых требований к целевой функции. Для нахождения экстремума по этому алгоритму не требуется вычислять производные целевой функции, функция может иметь разрывы как первого, так и второго рода или вообще представлять собой набор экспериментальных данных.

Алгоритм прост по своей сути и может быть легко запрограммирован.

При удачном подборе параметров, сходимость алгоритма довольно высока.

К главному недостатку метода следует отнести необходимость расчета целевой функции для произвольной точки пространства решений, что, в случае сложной целевой функции, может привести к недопустимым затратам машинного времени в процессе поиска решения.

Нами разработан алгоритм, позволяющий обойти эту проблему. Описанию этого алгоритма и посвящена настоящая статья.

Модифицированный алгоритм метода роя частиц. Существуют многочисленные модификации метода роя частиц. Чаще всего улучшения работы алгоритма добиваются путем соответствующего подбора параметров. мы предлагаем другой подход. Как было указано выше, одним из недостатков метода роя частиц является необходимость многократного вычисления значений целевой функции. Для упрощения вычислительного процесса и ускорения вычислений мы предлагаем разделить алгоритм на два этапа.

На первом этапе целевая функция заменяется ее упрощенной моделью в соответствии с разработанным нами подходом, описанным в работе [5]. Это позволяет существенно сократить объем вычислений и быстро локализовать область искомого экстремума.

На втором этапе мы возвращаемся к исходной целевой функции и ищем решение задачи в области, локализованной на первом этапе.

Таким образом, предлагаемый нами алгоритм будет состоять их следующих шагов:

1) построение модели целевой функции с использованием аддитивного или мультипликативного алгоритмов [5];
2) генерация роя из *n* частиц, случайным образом разбросанных по области поиска решения;
3) поиск решения с использованием классического алгоритма метода роя частиц и модели целевой функции;
4) определение границ области локализации решения;
5) генерация роя из *n* частиц, случайным образом разбросанных по найденной на предыдущем шаге области локализации решения;
6) поиск решения с использованием классического алгоритма метода роя частиц и исходной целевой функции;
7) если выполняется критерий останова, то переход к шагу 8, иначе – переход к шагу 6;
8) вывод координат положения лучшей частицы как результата решения оптимизационной задачи;
9) конец.

В соответствии с предложенным алгоритмом нами было написано консольное приложение на языке программирования C++, позволившее провести ряд численных экспериментов и оценить эффективность предлагаемого подхода.

Численные эксперименты. Оценка качества работы алгоритмов оптимизации осуществляется, как правило, путем проверки на специальных наборах тестовых функций. Такие наборы легко найти в Интернете. Мы воспользуемся виртуальной библиотекой тестовых функций и наборов данных, представленной S. Surjanovic и D. Bingham из университета Симона Файзера (Канада), которая расположена по адресу http://www.sfu.ca/~ssurjano.

При проведении экспериментов будем использовать следующие параметры роя. Число частиц в рое – 20, число итераций – 10.

Функция Растригина $f(x1, x2) = 20 + [x1^2 - 10 \cdot \cos(2 \cdot \pi \cdot x1)] + [x2^2 - 10 \cdot \cos(2 \cdot \pi \cdot x2)]$**.** Функция Растригина имеет несколько локальных минимумов. Функция сильно мультимодальная, но локальные минимумы регулярно распределены по области определения функции. На рисунке 1 показан график двумерной функции Растригина. Мы также представляем таблицу 1 с координатами первых десяти частиц роя, рассчитанными на первой, третьей и пятой итерациях.

Глобальный минимум функции имеет координаты [0, 0] и был найден на пятой итерации. Исходное положение частиц и изменение их положения в процессе работы программы иллюстрируется рисунками 2–5.

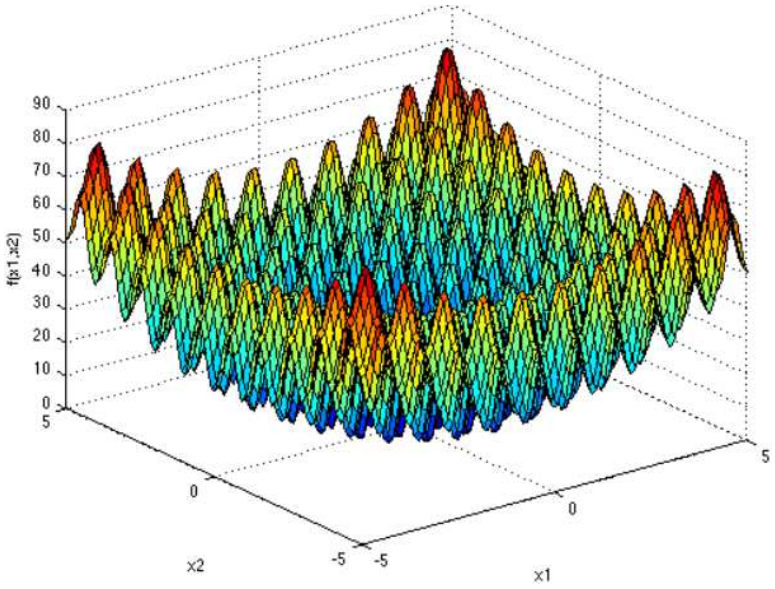

Рисунок 1. – Функция Растригина[2]

Таблица 1. – Изменение положения частиц в процессе работы программы. Целевая функция – функция Растригина

	Первые десять частиц роя										
*x*1	4,00	-1,00	1,00	1,00	-3,00	5,00	-3,00	-2,00	0,00	2,00	Исходное расположение частиц
*x*2	-5,00	1,00	-1,00	-4,00	4,00	5,00	-5,00	0,00	3,00	5,00	
*x*1	-2,00	0,50	-0,50	-0,50	1,50	-2,50	1,50	1,00	0,00	-1,00	Первая итерация
*x*2	1,00	-2,00	-1,00	0,50	-3,50	-4,00	1,00	-1,50	-3,00	-4,00	
*x*1	1,00	-0,25	0,25	0,25	-0,75	1,25	-0,75	-0,50	0,00	0,50	Третья итерация
*x*2	-2,00	-0,50	-1,00	-1,75	0,25	0,50	-2,00	-0,75	-1,03	0,50	
*x*1	0,25	-0,06	0,06	0,06	-0,19	0,31	-0,19	-0,13	0,00	0,13	Пятая итерация
*x*2	-0,50	-0,13	-0,25	-0,44	0,06	0,13	-0,50	-0,19	0,00	0,13	

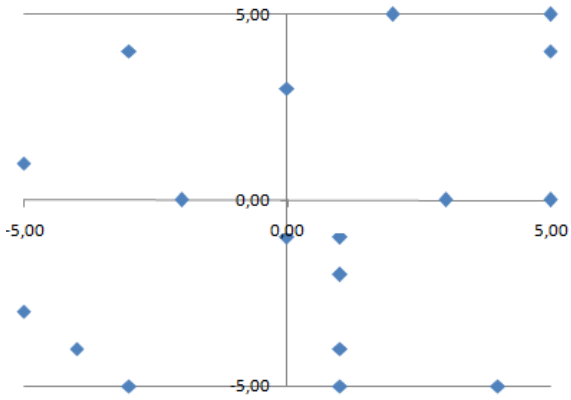

Рисунок 2. – Исходное положение частиц

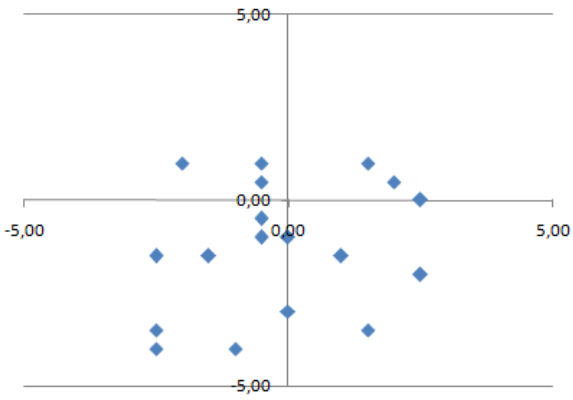

Рисунок 3. – Положение частиц после 1-й итерации

[2] Virtual Library of Simulation Exoeriments [Electronic resource]. URL: https://www.sfu.ca/~ssurjano/index.html.

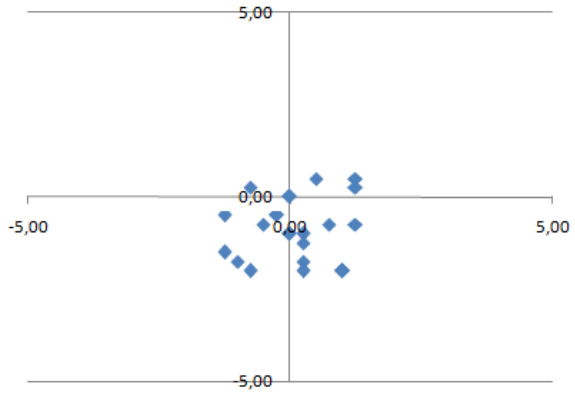

Рисунок 4. – Положение частиц после 3-й итерации

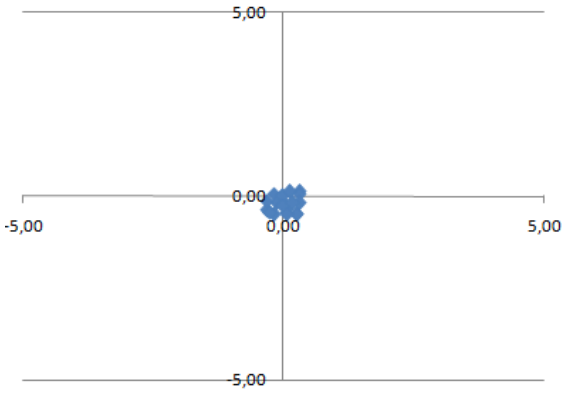

Рисунок 5. – Положение частиц после 5-й итерации

Функция Химмельблау $f(x1,x2)=(x1^2+x2-11)^2+(x1^2+x2^2-7)^2$ [3]**.** Это мультимодальная функция двух переменных. Имеет четыре равнозначных локальных минимума

$$f(3,2)=0;$$
$$f(-2,805118...,3,131312...)=0;$$
$$f(-3,779310...,-3,283186...)=0;$$
$$f(3,584428...,-1,848126...)=0,$$

локальный максимум с координатами

$$f(-0,270845...,-0,923039...)=181,716...$$

Изменяя начальные параметры расчета, можно найти все четыре локальных минимума. В таблице 2 приведены результаты расчета для одного из анализируемых вариантов.

Таблица 2. – Изменение положения частиц в процессе работы программы.
Целевая функция – функция Химмельблау

	Первые десять частиц роя										
*x*1	11,00	37,00	–31,00	–45,00	–23,00	14,00	–23,00	27,00	–21,00	32,00	Исходное расположение частиц
*x*2	–34,00	–7,00	21,00	–10,00	10,00	47,00	–30,00	14,00	–1,00	–24,00	
*x*1	–4,00	2,50	–14,50	–18,00	–12,50	–3,25	–12,50	0,00	–12,00	1,25	Первая итерация
*x*2	–9,25	–2,50	4,50	–3,25	1,75	11,00	–8,25	2,75	–1,00	–6,75	
*x*1	1,81	3,44	–0,81	–1,69	–0,31	2,00	–0,31	2,81	–0,19	3,13	Третья итерация
*x*2	–1,00	0,69	2,44	0,50	1,75	4,06	–0,75	2,00	1,06	–0,38	
*x*1	2,56	2,97	1,91	1,69	2,03	2,61	2,03	2,81	2,06	2,89	Пятая итерация
*x*2	1,25	1,67	2,11	1,63	1,94	2,52	1,31	2,00	1,77	1,41	
*x*1	3,02	3,04	2,98	2,96	2,98	3,02	2,98	3,03	2,99	3,04	Девятая итерация
*x*2	1,91	1,94	1,96	1,93	1,95	1,99	1,91	1,96	1,94	1,92	

[3] Virtual Library of Simulation Exoeriments [Electronic resource]. URL: https://www.sfu.ca/~ssurjano/index.html.

Функция Била $f(x1,x2)=(1{,}5-x1+x1\cdot x2)^2+(2{,}25-x1+x1\cdot x2^2)^2+(2{,}625-x1+x1\cdot x2^3)^2$ [4]**.** Это мультимодальная функция с острыми пиками в углах области определения. Глобальный минимум функции $f(3{,}0{,}5)=0.$

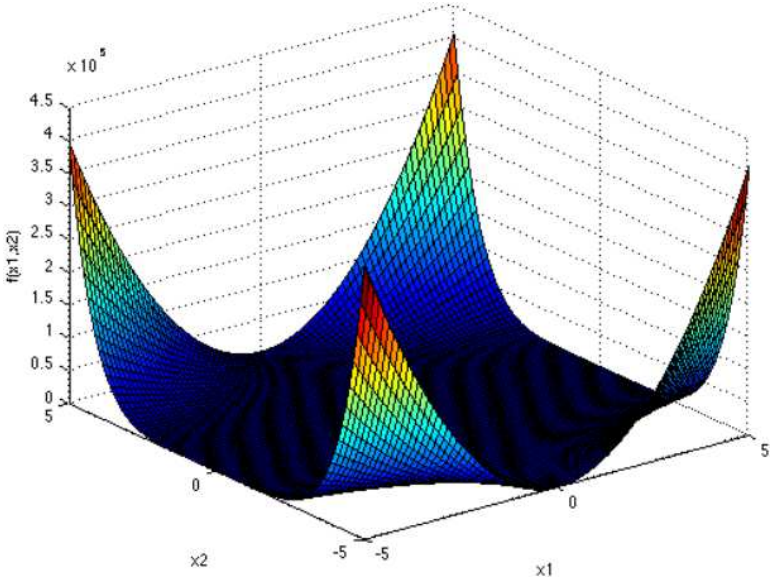

Рисунок 6. – Функция Била[5]

В таблице 3 приведены координаты первых десяти частиц для первой, третьей и пятой итераций.

Таблица 3. – Изменение положения частиц в процессе работы программы. Целевая функция – функция Била

	Первые десять частиц роя										
*x*1	0,00	–2,00	3,00	–3,00	2,00	–2,00	1,00	3,00	4,00	3,00	Исходное расположение частиц
*x*2	1,00	–1,00	1,00	–3,00	1,00	3,00	–1,00	4,00	0,00	1,00	
*x*1	1,50	2,50	0,00	3,00	0,50	2,50	1,00	0,00	–0,50	0,00	Первая итерация
*x*2	–2,00	–1,00	–2,00	0,00	–2,00	–3,00	–1,00	–3,50	–1,50	–2,00	
*x*1	3,00	3,25	2,63	3,38	2,75	3,25	2,88	2,63	2,50	2,63	Третья итерация
*x*2	0,25	0,50	0,25	0,75	0,25	0,00	0,50	–0,13	0,38	0,25	
*x*1	2,91	2,97	2,81	3,00	2,84	2,97	2,88	2,81	2,78	2,81	Пятая итерация
*x*2	0,44	0,50	0,44	0,56	0,44	0,38	0,50	0,34	0,47	0,44	

Заключение.

1. Полученные результаты подтверждают высокую эффективность разработанного алгоритма.
2. Алгоритм обладает высокой скоростью сходимости, что делает возможным получение приемлемого решения даже при малом числе итераций.
3. Имеющиеся параметры позволяют управлять вычислительным процессом и находить все экстремумы целевой функции.
4. Следующим шагом в развитии алгоритма может стать применение при его построении идей эволюционного программирования – разбиение исходного роя частиц на подрои и организация конкурентной борьбы между подроями.

[4] Virtual Library of Simulation Exoeriments [Electronic resource]. URL: https://www.sfu.ca/~ssurjano/index.html.
[5] Там же.

ЛИТЕРАТУРА

1. Kennedy, J. Particle swarm optimization / J. Kennedy, R. Eberhart // Proceedings of ICNN'95 – International Conference on Neural Networks. – 1995. – Vol. 4. – P. 1942–1948. – DOI: 10.1109/ICNN.1995.488968.
2. Kennedy, J. Swarm Intelligence / J. Kennedy, R. C. Eberhart. – Morgan Kaufmann, 2001. – 541 p.
3. Poli, R. An analysis of publications on particle swarm optimisation applications / R. Poli // Technical Report CSM-469 / Department of Computer Science, University of Essex, UK. – 2007. – 57 p.
4. Poli, R. Analysis of the Publications on the Applications of Particle Swarm Optimisation / R. Poli // J. of Artificial Evolution and Applications. – 2008. – P. 1–10. – DOI: 10.1155/2008/685175.
5. Оськин, А. Ф. Алгоритмы приближения элементов матрицы компонентами двух векторов / А. Ф. Оськин // Вестн. Полоц. гос. ун-та. Сер. С, Фундам. науки. – 2004, – № 4. – С. 73–76.

REFERENCES

1. Kennedy, J., & Eberhart, R. (1995) Particle swarm optimization. *Proceedings of ICNN'95 – International Conference on Neural Networks: Vol. 4* (1942–1948). DOI: 10.1109/ICNN.1995.488968.
2. Kennedy, J., & Eberhart, R.C. (2001) *Swarm Intelligence*. Morgan Kaufmann.
3. Poli, R. (2007) An analysis of publications on particle swarm optimisation applications. *Technical Report CSM-469.* Department of Computer Science, University of Essex, UK.
4. Poli, R. (2008) Analysis of the Publications on the Applications of Particle Swarm Optimisation. *Journal of Artificial Evolution and Applications*, 1–10. DOI: 10.1155/2008/685175.
5. Os'kin, A. F. (2004) Algoritmy priblizheniya elementov matritsy komponentami dvukh vektorov [Algorithms for Approximation of Matrix Elements by Components of Two Vectors] *Vestnik Polotskogo gosudarstvennogo universiteta. Seriya C, Fundamental'nye nauki [Herald of Polotsk State University. Series C. Fundamental sciences]*, (4), 73–76. (In Russ., abstr. in Engl.).

Поступила 22.03.2022

NUMERICAL OPTIMIZATION ALGORITHM AND PROGRAM IMPLEMENTING THE PARTICLE SWAR OPTIMIZATION

A. OSKIN, D. OSKIN

One of the numerical optimization methods is considered, which implements the so-called particle swarm method. A modification of the method based on the division of the iterative calculation process into two stages is proposed. To speed up calculations and reduce their complexity. at the first stage, the objective function is replaced by a simplified model, which allows you to quickly determine the approximate area of extremum localization. The final solution is sought in the found localization area, using the original objective function. A console application that implements the algorithm is described and the results of numerical experiments performed using this application are presented.

Keywords: *numerical optimization, particle swarm method, software implementation of the particle swarm method.*