УДК 519.688
https://doi.org/10.47533/2024.1606-146X.54

А. К. КАЙРАКБАЕВ

Учреждение «Баишев университет», Актобе, Казахстан
E-mail: kairak@mail.ru

АНАЛИЗ И ОЦЕНКА ЭФФЕКТИВНОСТИ ПРОГРАММНЫХ КОДОВ МЕТОДОВ РОЯ ЧАСТИЦ И КОЛОНИИ МУРАВЬЕВ ДЛЯ ПОЛУНЕПРЕРЫВНЫХ СНИЗУ ФУНКЦИЙ

Кайракбаев Аят Крымович – профессор, профессор Высшей школы Социально-технических наук Учреждения «Баишев университет», Актобе, Казахстан.
E-mail: kairak@mail.ru

В настоящей работе проведен сравнительный анализ двух стохастических методов глобальной оптимизации: метода роя частиц (Particle Swarm Optimization) и метода колонии муравьев (Ant Colony Optimization). Анализ и оценка эффективности проводились с помощью таких параметров, как число вызовов целевой функции, скорость сходимости, ресурсы, вычислительная эффективность, чувствительность к параметрам, устойчивость к начальным условиям. В статье представлены программные коды данных методов в среде Python, предназначенные для вычисления глобального экстремума полунепрерывных снизу функций. Были использованы известные тестовые функции для построения (путем склеивания) полуненпрерывных функций при различных значениях входных параметров.

Ключевые слова: *метод роя частиц, метод колонии муравьев, глобальная оптимизация, число вызовов целевой функции, скорость сходимости, ресурсы, вычислительная эффективность.*

A. K. KAIRAKBAEV

Institution «Baishev University», Aktobe, Kazakhstan
E-mail: kairak@mail.ru

ANALYSIS AND EVALUATION OF THE EFFECTIVENESS OF PROGRAM CODES FOR PARTICLE SWARM AND ANT COLONY METHODS FOR LOWER SEMI-CONTINUOUS FUNCTIONS

Kairakbaev Ayat Krymovich – Professor, Professor of the Higher School of Social and Technical Sciences of the «Baishev University» Institution, Aktobe, Kazakhstan.
E-mail: kairak@mail.ru

In this paper, a comparative analysis of two stochastic methods of global optimization is carried out: the Particle Swarm Optimization method and the Ant Colony Optimization method. The analysis and evaluation of efficiency were carried out using parameters such as the number of calls to the objective function, the rate of convergence, resources, computational efficiency, sensitivity to parameters, and resistance to initial conditions. The article presents the program codes of these methods in the Python environment, designed to calculate the global extremum of semi-continuous functions from below. Well-known test functions were used to construct (by gluing) semi-continuous functions at different values of input parameters.

Keywords: *particle swarm method, ant colony method, global optimization, number of calls to the objective function, convergence rate, resources, computational efficiency.*

А. К. ҚАЙРАҚБАЕВ

«Баишев университеті» мекемесі, Ақтөбе, Қазақстан
E-mail: kairak@mail.ru

ТӨМЕННЕН ЖАРТЫЛАЙ ҮЗДІКСІЗ ФУНКЦИЯЛАР ҮШІН БӨЛШЕКТЕР ҮЙІРІ ӘДІСІ МЕН ҚҰМЫРСҚАЛАР КОЛОНИЯСЫ ӘДІСТЕРІНІҢ БАҒДАРЛАМАЛЫҚ КОДТАРЫНЫҢ ТИІМДІЛІГІН ТАЛДАУ ЖӘНЕ БАҒАЛАУ

Кайракбаев Аят Крымович – профессор, «Баишев университеті» мекемесінің әлеуметтік-техникалық ғылымдар Жоғары мектебінің профессоры, Ақтобе, Қазақстан.

E-mail: kairak@mail.ru

Бұл жұмыста жаһандық оңтайландырудың екі стохастикалық әдісіне салыстырмалы талдау жүргізілді: бөлшектер тобы әдісі (Particle Swarm Optimization) және құмырсқалар колониясы әдісі (Ant Colony Optimization). Тиімділікті талдау және бағалау мақсатты функция қоңырауларының саны, конвергенция жылдамдығы, ресурстар, есептеу тиімділігі, параметрлерге сезімталдық, бастапқы жағдайларға төзімділік сияқты параметрлер арқылы жүргізілді. Мақалада Python ортасындағы жартылай үздіксіз функциялардың жаһандық экстремумын есептеуге арналған әдістердің бағдарламалық деректер кодтары берілген. Кіріс параметрлерінің әртүрлі мәндерінде жартылай үздіксіз функцияларды құру (желімдеу арқылы) үшін белгілі сынақ функциялары қолданылды.

Түйін сөздер: *бөлшектерді жинау әдісі, құмырсқалар колониясы әдісі, ғаламдық оңтайландыру, мақсатты функция қоңырауларының саны, конвергенция жылдамдығы, ресурстар, есептеу тиімділігі.*

Введение. В настоящее время известны две категории основных методов глобальной оптимизации [1]: детерминированные и стохастические. Детерминированным методам оптимизации [2-4] относятся такие методы, как градиентный спуск, метод Ньютона, метод сопряженных градиентов и т.д. [5,6]. Основным преимуществом детерминированных методов является способность точно находить локальные экстремумы гладких функций. А к недостаткам можно отнести их чувствительность к начальным данным, когда целевая функция содержит неопределенные компоненты, а также в задачах с шумом [7].

Стохастические методы [8-10] позволяют находить приближенные решения оптимизационной задачи используя случайные элементы или вероятностные процессы в своих алгоритмах. К таким методам относятся, к примеру, генетические алгоритмы, метод роя частиц, метод колонии муравьев [11], и т.д. Эти методы более эффективны при решении задач, где функция цели шумная или неопределенная, а также в случаях, когда вычислительные ресурсы ограничены, а объемы данных большие и размерность пространства переменных велика. При этом основным пре-

имуществом их является способность приближенно находить глобальные экстремумы.

Какую бы задачу глобальной оптимизации исследователь не решал, рано или поздно он столкнется проблемой выбора наиболее подходящего выбора метода из всех существующих [12,13]. Другими словами, ему необходимо будет выяснить, какой метод быстрее сходится к оптимальному, у какого метода решение более точное, насколько сам метод чувствителен к исходным параметрам, насколько метод устойчив к начальным данным и какие вычислительные ресурсы требует. Поэтому исследования, посвященные сравнительному анализу и оценке эффективности методов оптимизации, являются актуальными.

В работе рассмотрены два стохастических метода: метод роя частиц и метод колонии муравьев [11]. Оба метода инспирированы коллективным поведением живых существ из окружающей среды при поиске оптимального решения. Метод роя частиц имитирует кооперативное поведение стаи птиц или рыб. В этом методе частицы двигаются в пространстве решений, последовательно обновляя свои скорости и позиции. А метод колонии муравьев имитирует поведение муравьев при поиске оптимального пути между муравейником и добычей муравьев. Каждый муравей метит свой путь феромонами, которые влияют на выбор пути другого муравья. Чем более путь успешен, тем больше он получает феромонов. Данный метод моделирует процесс распределения феромонов на путях и использует их для выбора оптимального пути. Оба метода использует информацию о предыдущих решениях для поиска наилучшего решения.

Программный код метода роя частиц

```
import numpy as np
def objective_function(x):
    # Целевая функция Ф(x)
    return sum(x**2)
class Particle:
    def __init__(self, num_dimensions, lower_bounds, upper_bounds):
            self.position = np.random.uniform(lower_bounds, upper_bounds)
            self.velocity = np.random.uniform(-1, 1, num_dimensions)
            self.best_position = np.copy(self.position)
            self.best_value = objective_function(self.position)
def particle_swarm_optimization(num_particles, num_iterations, lower_bounds, upper_bounds):
num_dimensions = len(lower_bounds)
    # Инициализация частиц
particles = [Particle(num_dimensions, lower_bounds, upper_bounds) for _ in range(num_particles)]
    # Инициализация глобального лучшего решения
global_best_position = None
global_best_value = float('inf')
    # Инициализация метрик
num_function_calls = 0
```

```
convergence_speed = 0.0
resources = 0.0
computational_efficiency = 0.0
sensitivity_to_parameters = 0.0
stability_to_initial_conditions = 0.0
      # Основной цикл
for iteration in range(num_iterations):
for particle in particles:
            # Обновление скорости и позиции частицы
inertia_weight = 0.5 # Инерционный вес
     cognitive_weight = 2.0 # Когнитивный вес
     social_weight = 2.0 # Социальный вес
r1 = np.random.rand(num_dimensions)
r2 = np.random.rand(num_dimensions)
particle.velocity = (inertia_weight * particle.velocity +
 cognitive_weight * r1 * (particle.best_position - particle.position) +
 social_weight * r2 * (global_best_position - particle.position))
particle.position += particle.velocity
            # Ограничение позиции в допустимых пределах
particle.position = np.clip(particle.position, lower_bounds, upper_bounds)
            # Вычисление значения целевой функции
current_value = objective_function(particle.position)
num_function_calls += 1
            # Обновление лучшего значения и позиции для частицы
if current_value < particle.best_value:
particle.best_value = current_value
particle.best_position = np.copy(particle.position)
            # Обновление глобального лучшего значения
if current_value < global_best_value:
 global_best_value = current_value
 global_best_position = np.copy(particle.position)
     # Обновление метрик
convergence_speed = 1.0 / (1.0 + np.abs(global_best_value - particle.best_value))
resources = num_particles * num_dimensions
computational_efficiency = global_best_value / num_function_calls
sensitivity_to_parameters = cognitive_weight / (cognitive_weight + social_weight)
stability_to_initial_conditions = np.std([particle.best_value for particle in particles])
     # Вывод результатов
print("Best Solution:", global_best_position)
print("Best Objective Value:", global_best_value)
print("Number of Function Calls:", num_function_calls)
print("Convergence Speed:", convergence_speed)
print("Resources:", resources)
print("Computational Efficiency:", computational_efficiency)
```

```
print("Sensitivity to Parameters:", sensitivity_to_parameters)
print("Stability to Initial Conditions:", stability_to_initial_conditions)
# Пример использования
num_particles = 10
num_iterations = 100
lower_bounds = np.array([0.0, 0.0]) # Пример для двух переменных
upper_bounds = np.array([1.0, 1.0])
particle_swarm_optimization(num_particles, num_iterations, lower_bounds, upper_
bounds)
```

Основные параметры, влияющие на работу программного кода следующие.

1) num_particles - количество частиц в рое, равное числу различных решений в каждой итерации. Обычно выбирается в диапазоне от 10 до 100 в зависимости от сложности задачи.

2) num_iterations - количество итераций алгоритма, равное числу повторений цикла обновлений положений и скоростей частиц. Может быть в диапазоне от 50 до 1000 в зависимости от желаемой длительности выполнения алгоритма.

3) lower_bounds – массив из нижних границ переменных.

4) upper_bounds - – массив из верхних границ переменных.

5) inertia_weight - инерционный вес, влияющий на сохранение частицами своей текущей скорости. Обычно варьируется от 0.1 до 1.0.

6) cognitive_weight - когнитивный вес, определяющий влияние лучшего собственного значения частицы на ее движение. Может принимать значения от 1 до 2.

7) social_weight - социальной вес, определяющий влияние глобального лучшего значения на движение частицы. Может принимать значения от 1 до 2.

8) r1 и r2 - случайные веса для обновления скорости частиц, используемые для добавления случайного элемента в обновление скорости. Обычно равны случайному числу в диапазоне от 0 до 1.

9) num_dimensions - количество переменных в целевой функции.

Программный код метода колонии муравьев

```
import numpy as np
def objective_function(x):
    # Целевая функция Ф(x)
return sum(x**2)
def ant_colony_optimization(num_ants, num_iterations, lower_bounds, upper_
bounds):
    # Инициализация параметров ACO
rho = 0.1
alpha = 1.0
beta = 2.0
Q = 1.0
pheromone_init = 0.01
num_dimensions = len(lower_bounds)
    # Инициализация феромона
```

```
 pheromone = np.full(num_ants, pheromone_init)
        # Инициализация начальных решений муравьев
    ants_solutions = np.random.uniform(lower_bounds, upper_bounds, size=(num_ants, num_dimensions))
        # Инициализация метрик
    num_function_calls = 0
    convergence_speed = 0.0
    resources = 0.0
    computational_efficiency = 0.0
    sensitivity_to_parameters = 0.0
    stability_to_initial_conditions = 0.0
        # Основной цикл
    for iteration in range(num_iterations):
    objective_values = np.apply_along_axis(objective_function, 1, ants_solutions)
    num_function_calls += num_ants
         # Обновление феромона на путях муравьев
    pheromone *= (1.0 - rho)
    pheromone += Q / objective_values
        # Выбор следующего решения для каждого муравья
    for ant in range(num_ants):
    probabilities = (pheromone ** alpha) * ((1.0 / objective_values) ** beta)
    probabilities /= probabilities.sum()
    selected_solution = np.random.choice(np.arange(len(ants_solutions)), p=probabilities)
    ants_solutions[ant] = ants_solutions[selected_solution]
        # Обновление метрик
    best_solution_index = np.argmin(objective_values)
    best_objective_value = objective_values[best_solution_index]
    convergence_speed = 1.0 / (1.0 + np.abs(objective_values - best_objective_value))
    resources = np.sum(pheromone)
    computational_efficiency = best_objective_value / num_function_calls
    sensitivity_to_parameters = alpha / (alpha + beta)
    stability_to_initial_conditions = np.std(objective_values)
        # Вывод результатов
    best_solution = ants_solutions[best_solution_index]
    print("Best Solution:", best_solution)
    print("Best Objective Value:", best_objective_value)
    print("Number of Function Calls:", num_function_calls)
    print("Convergence Speed:", convergence_speed)
    print("Resources:", resources)
    print("Computational Efficiency:", computational_efficiency)
    print("Sensitivity to Parameters:", sensitivity_to_parameters)
    print("Stability to Initial Conditions:", stability_to_initial_conditions)
    # Пример использования
```

```
num_ants = 10
num_iterations = 100
lower_bounds = np.array([0.0, 0.0]) # Пример для двух переменных
upper_bounds = np.array([1.0, 1.0])
ant_colony_optimization(num_ants, num_iterations, lower_bounds, upper_bounds)
```

Далее приведем краткое описание параметров метода колонии муравьев.

1) Параметр num_ants указывает на количество муравьев в колонии. Он определяет, сколько различных решений будет исследовано в каждой итерации. Обычно выбирается в диапазоне от 5 до 100 в зависимости от сложности задачи.

2) Переменная num_iterations означает число итераций алгоритма. Определяет, сколько раз будет повторяться основной цикл обновления феромона и перемещения муравьев. Может быть установлен в диапазоне от 50 до 1000 в зависимости от желаемой длительности выполнения алгоритма.

3) rho - коэффициент испарения феромона, определяет скорость испарения феромона на путях муравьев. Обычно варьируется от 0.1 до 0.9.

4) alpha - вес феромона, определяющий влияние феромона на выбор следующего решения.

5) beta - вес привлекательности, определяющий влияние привлекательности решения на выбор следующего решения. Значения alpha и beta могут быть в диапазоне от 0.1 до 2.0.

6) Q - количество феромона выделяемое каждым муравьем. Определяет, сколько феромона будет добавлено на путь, который выбрал муравей. Зависит от масштаба задачи, но обычно находится в диапазоне от 1 до 100.

7) pheromone_init - начальная концентрация феромона на путях.

8) lower_bounds - массив, представляющий нижние границы для каждой переменной.

9) upper_bounds - массив, представляющий верхние границы для каждой переменной.

Выводы. С помощью приведенных программных кодов проводилось большое количество численных вычислительных экспериментов на полунепрерывных функциях, склеенных из тестовых функций. В результате выполнения приведенные программные коды выдают такие показатели, как число вызовов целевой функции, скорость сходимости, ресурсы, вычислительную эффективность алгоритма и глобальный минимум.

Таким образом, метод роя частиц имеет преимущества по скорости сходимости и более быстрой адаптации к изменяющимся условиям. К недостаткам можно отнести высокую зависимость от выбора параметров алгоритма и потерю точности при решении задач с большой размерностью. Метод колонии муравьев имеет преимущества в виде высокой точности нахождения глобального минимума при решении задач с большой размерностью. Однако, метод колонии муравьев требует сложных вычислительных операций и имеет низкую скорость сходимости, а также настройки большого количества параметров.

ЛИТЕРАТУРА

1 Floudas, Christodoulos A., and Panos M. Pardalos. «Recent advances in global optimization.» (2014).

2 Cavazzuti M., Cavazzuti M. Deterministic optimization //Optimization methods: From theory to design scientific and technological aspects in mechanics. – 2013. – С. 77-102.

3 Horst, Reiner, and Hoang Tuy. Global optimization: Deterministic approaches. Springer Science & Business Media, 2013.

4 Lin, Ming-Hua, Jung-Fa Tsai, and Chian-Son Yu. «A review of deterministic optimization methods in engineering and management.» Mathematical Problems in Engineering 2012 (2012).

5 Nazareth, John L. «Conjugate gradient method.» Wiley Interdisciplinary Reviews: Computational Statistics 1.3 (2009): 348-353.

6 Adams, Loyce M., and John Lawrence Nazareth, eds. Linear and nonlinear conjugate gradient-related methods. Vol. 85. Siam, 1996.

7 Acar E. et al. Modeling, analysis, and optimization under uncertainties: a review //Structural and Multidisciplinary Optimization. – 2021. – Т. 64. – №. 5. – С. 2909-2945.

8 Marti K. et al. Stochastic optimization methods. – Berlin : Springer, 2008. – Т. 2.

9 Zabinsky, Zelda B. «Stochastic methods for practical global optimization.» Journal of Global Optimization 13 (1998): 433-444.

10 Zhigljavsky, Anatoly, and Antanasz Zilinskas. Stochastic global optimization. Vol. 9. Springer Science & Business Media, 2007.

11 Sivanandam, S. N., et al. «Introduction to particle swarm optimization and ant colony optimization.» Introduction to genetic algorithms (2008): 403-424.

12 Агасиев, Т. А., and А. П. Карпенко. «Современные техники глобальной оптимизации. Обзор.» Информационные технологии 24.6 (2018): 370-386.

13. Stork, J., Eiben, A.E. & Bartz-Beielstein, T. A new taxonomy of global optimization algorithms. Nat Comput 21, 219–242 (2022).

14. Li, Xiaodong, et al. «Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization.» gene 7 (2013): 33.

REFERENCES

1 Floudas, Christodoulos A., and Panos M. Pardalos. "Recent advances in global optimization." (2014).

2. Cavazzuti M., Cavazzuti M. Deterministic optimization //Optimization methods: From theory to design scientific and technological aspects in mechanics. – 2013. – С. 77-102.

3 Horst, Reiner, and Hoang Tuy. Global optimization: Deterministic approaches. Springer Science & Business Media, 2013.

4 Lin, Ming-Hua, Jung-Fa Tsai, and Chian-Son Yu. "A review of deterministic optimization methods in engineering and management." Mathematical Problems in Engineering 2012 (2012).

5 Nazareth, John L. "Conjugate gradient method." Wiley Interdisciplinary Reviews: Computational Statistics 1.3 (2009): 348-353.

6 Adams, Loyce M., and John Lawrence Nazareth, eds. Linear and nonlinear conjugate gradient-related methods. Vol. 85. Siam, 1996.

7 Acar E. et al. Modeling, analysis, and optimization under uncertainties: a review //Structural and Multidisciplinary Optimization. – 2021. – Т. 64. – №. 5. – С. 2909-2945.

8 Marti K. et al. Stochastic optimization methods. – Berlin : Springer, 2008. – Т. 2.

9 Zabinsky, Zelda B. "Stochastic methods for practical global optimization." Journal of Global Optimization 13 (1998): 433-444.

10 Zhigljavsky, Anatoly, and Antanasz Zilinskas. Stochastic global optimization. Vol. 9. Springer Science & Business Media, 2007.

11 Sivanandam, S. N., et al. “Introduction to particle swarm optimization and ant colony optimization.” Introduction to genetic algorithms (2008): 403-424.

12 Агасиев, Т. А., and А. П. Карпенко. “Современные техники глобальной оптимизации. Обзор.” Информационные технологии 24.6 (2018): 370-386.

13 Stork, J., Eiben, A.E. & Bartz-Beielstein, T. A new taxonomy of global optimization algorithms. Nat Comput 21, 219–242 (2022).

14 Li, Xiaodong, et al. “Benchmark Functions for the CEC’2013 Special Session and Competition on Large-Scale Global Optimization.” gene 7 (2013): 33.