Chernobyl disaster optimization

Примакова Е.Е.

гр. 9301

1. Введение

Chernobyl Disaster Optimizer (CDO) - это метаэвристический алгоритм, вдохновленный поведением ядерной радиации во время Чернобыльской катастрофы. Он имитирует движение α, β и γ частиц, которые распространяются от центра взрыва (области наибольшего давления) и «атакуют» человеческие цели (области наименьшего давления). При этом алгоритм, также учитывает физические свойства частиц, такие как: скорость распространения и сособность проходить через те или иные материалы. Каждый тип частицы (α, β и γ) по сути своей является уникальным поисковым агентом.

Так γ частицы, являясь наиболее быстрыми и "проникающими", играют роль агентов глобального поиска - наиболее быстро исследуют пространство поиска, при этом с не очень высокой точностью. β частицы - агенты балансирующие между глобальным и локальным поиском, в то время как α частицы - самые медленные и с наихудшей способность проходить через материалы, являются агентвми локального поиска.

Как и в случае с реальным радиоактивным излучением, β и γ частицы проходят сквозь человеческое тело, в то время как α останавливаются. Поэтому поисковым агентом, который приходит к глобальному миимуму являются именно α частицы.

Также следует отметить, что поисковые агенты зависимы между собой по аналогии с PSO - все притягиваются к текущему лучшему решению, что обеспечивает сходимость и эффективное использование свойств поисковых агентов.

2. Алгоритм и тестовые функции

2.1. Описание алгоритма

Как уже говорилось, после ядерного взрыва излучение состит из α, β и γ частиц. Эти частицы летят от точки взрыва (область высокого давления) до тех пор, пока не достигнут человеческих территорий (область низкого давления), где и произойдет катастрофа. Алгоритм предролагает, что жертвы (люди) идут, и эти частицы атакуют их в одно и тоже время, с разной скоростью (Табл. 1).

Тип частицы	Скорость (s)
α	$0.16 \cdot 10^5$
β	$2.7 \cdot 10^5$
γ	$3 \cdot 10^5$

Таблица 1: Скорость частиц

Скорость движения людей "бегущих" от радиации линейно снижается с каждой итерацией t, иными словами, в соответствии с (1), на последней итерации алгоритма T все умрут.

$$WS_h = 3 - 3\frac{t}{T}, \tag{1}$$

Изначально создаются n агентов, которые представляют собой частицы трёх типов. Их начальные позиции распределяются случайным образом по всему пространству поиска. Скорости частиц (s^γ, s^β, s^α) задаются в соответствии с их физическими характеристиками. Например, скорость γ-частиц является самой высокой, что делает их эффективными в глобальном поиске, тогда как α-частицы действуют медленно, выполняя локальную оптимизацию.

Обновление позиций частиц осуществляется на каждой итерации на основе следующих формул:

$$v_{i,j}(t+1) = \sum_{p \in \{\gamma,\beta,\alpha\}} \left(\frac{r \cdot A \cdot P_{i,j} - P_j^p}{s^p \cdot \text{rand}(0,1)} - WS_h \right), \tag{2}$$

$$P_{i,j}(t+1) = P_{i,j}(t) + v_{i,j}(t+1), \tag{3}$$

где $v_{i,j}$ — скорость частицы, $P_{i,j}$ — положение частицы, A — радиус действия частиц, WS_h — скорость жертвы, а r — случайный коэффициент.

Каждая частица оценивается по значению целевой функции f. Если текущая пригодность частицы лучше, чем у её типа (например, у α-частиц), то обновляются наилучшие известные значения для данного типа частиц.

Процесс повторяется до тех пор, пока не будет достигнуто максимальное количество итераций T или не будет удовлетворено условие остановки. Алгоритм поддерживает равновесие между глобальным и локальным поиском благодаря сочетанию различных типов частиц и их взаимодействию через обновление скоростей.

Ниже приведены основные этапы алгоритма CDO:

1. **Инициализация.** Создание агентов с случайными начальными позициями.
2. **Обновление скоростей и позиций.** Расчёт новых скоростей и позиций с использованием физических моделей распространения частиц (1-3).
3. **Оценка.** Вычисление значений целевой функции и обновление лучших значений.
4. **Конвергенция.** Завершение работы алгоритма при достижении заданных условий.

Основной принцип CDO заключается в использовании разных типов частиц, чтобы обеспечить эффективный баланс между исследованием пространства и уточнением решений .

Ниже приведен псевдокод рассматриваемого метода оптимизации. Входные параметры алгоритма указаны в псевдокоде.

В пердставленном псевдокоде функция случайных чисел `rand` принимает первым аргументом размерность вектора случайных чисел, вторым аргументом - пределы в которых выбирается случайное число.

Algorithm 1: Chernobyl Disaster Optimizer (CDO)

Входные данные:
f ; /* Целевая функция */
d ; /* Размерность проблемы */
n ; /* Число частиц */
T ; /* Максимальная итерация */
$[l, u]$; /* Границы начального распределения частиц */
Результат:
x_{min} ; /* Точка минимума */
f_{min} ; /* Значение функции в $x_{\min}$ */
Алгоритм:
/* Константы скорости распространения частиц */
$s^{\gamma},\ sf^{\gamma} \leftarrow 3 \cdot 10^5,\ 1;$
$s^{\alpha},\ sf^{\alpha} \leftarrow 0.16 \cdot 10^5,\ 0.25;$
$s^{\beta},\ sf^{\beta} \leftarrow 2.7 \cdot 10^5,\ 0.5;$
/* Инициализация положений частиц в пространстве */
$P_i \leftarrow \texttt{rand}(d,\ [l, u]) \mid i \in [1 \dots n];$
/* Основной цикл */
for $t \leftarrow 1$ ***to*** T **do**
  for $i \leftarrow 1$ ***to*** n **do**
    $fitness \leftarrow f(P_i);$
    if *fitness* $< \alpha Score$ **then**
      $\alpha Score \leftarrow$ *fitness*;
      $P^{\alpha} \leftarrow P_i;$
    else if *fitness* $< \beta Score$ **then**
      $\beta Score \leftarrow$ *fitness*;
      $P^{\beta} \leftarrow P_i;$
    else if *fitness* $< \gamma Score$ **then**
      $\gamma Score \leftarrow$ *fitness*;
      $P^{\gamma} \leftarrow P_i;$
    end
  end
  $WS_h \leftarrow 3 - 3(\frac{t}{T})$; /* Скорость ходьбы человека */
  /* Обновление положений частиц в пространстве */
  for $i \leftarrow 1$ ***to*** n **do**
    $\nu_j \leftarrow 0 \mid j \in [1 \dots d]$; /* Инициализация градиентов */
    for $j \leftarrow 1$ ***to*** d **do**
      foreach p ***in*** $[\gamma, \alpha, \beta]$ **do**
        $S \leftarrow sf^p \cdot \log(\texttt{rand}(1,\ [1, s^p]))$; /* Скорость частицы */
        $x_h \leftarrow \pi \cdot \texttt{rand}(1,\ [0, 1])^2$; /* Радиус зоны ходьбы людей */
        $\rho \leftarrow \frac{x_h}{S} - WS_h \cdot \texttt{rand}(1,\ [0, 1]))$; /* Распространение частицы */
        $A \leftarrow \pi \cdot \texttt{rand}(1,\ [0, 1])^2$; /* Радиус распространения частицы */
        $\Delta \leftarrow \left| A \cdot P_{i,j} - P_j^p \right|;$
        $\nu_j \leftarrow \nu_j + P_{i,j} - \rho \cdot \Delta$; /* Вектор скорости частицы */
      end
    end
    $P_i = \nu \oslash 3;$
  end
end
$f_{min}, x_{min} \leftarrow \alpha Score,\ P^{\alpha};$
return x_{min}, f_{min}

2.2. Тестовые проблемы

2.2.1. Проблеммы с множественным локальным минимумом

Функции этой категории характеризуются наличием множества локальных минимумов, что делает оптимизацию сложной для алгоритмов, основанных на градиентных методах, которые могут легко

застрять в локальных минимумах. Эвристические и метаэвристические подходы, такие как генетические алгоритмы, оптимизация роя частиц и моделируемый отжиг, обычно лучше подходят для таких функций, поскольку они исследуют пространство решений более глобально.

Ackley Function Функция `Ackley` широко используется для тестирования алгоритмов оптимизации благодаря множеству локальных минимумов и единственному глобальному минимуму. Высокая частота колебаний мешает градиентным методам, в то время как метаэвристические алгоритмы превосходят их благодаря способности исследовать различные области пространства решений.

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e, \tag{4}$$

$$-5 \le x_i \le 5, \quad x_{\min} = (0, \ldots, 0), \quad f_{\min} = 0.$$

Eggholder Function Функция `Eggholder` известна своим сложным ландшафтом с многочисленными крутыми долинами и хребтами. Резкие переходы градиента делают ее сложной для квазиградиентных методов. Такие алгоритмы, как дифференциальная эволюция и роевые методы, часто оказываются более эффективными.

$$f(x_1, x_2) = -(x_2 + 47)\sin\left(\sqrt{|x_2 + x_1/2 + 47|}\right) - x_1\sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right), \tag{5}$$

$$-512 \le x_1, x_2 \le 512, \quad x_{\min} \approx (512, 404.2319), \quad f_{\min} \approx -959.6407.$$

2.2.2. Проблемы "чаши"

Эти функции гладкие и унимодальные, с одним глобальным минимумом, который лежит в параболической области. Они хорошо подходят для градиентных и квазиньютоновских методов благодаря своей выпуклой природе.

Sphere Function `Sphere` это простой квадратичный тест, используемый для оценки алгоритмов оптимизации. Его гладкая поверхность делает его идеальным для алгоритмов, использующих информацию о градиенте.

$$f(x) = \sum_{i=1}^{d} x_i^2, \tag{6}$$

$$-5.12 \le x_i \le 5.12, \quad x_{\min} = (0, \ldots, 0), \quad f_{\min} = 0.$$

Trid Function Функция `Trid` создана для того, чтобы бросить вызов алгоритмам оптимизации с неквадратичной формой чаши. Хотя она остается унимодальной, нелинейность требует квазиньютоновских методов или продвинутых градиентных оптимизаторов для эффективного сближения.

$$f(x) = \sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1}, \tag{7}$$

$$-d^2 \le x_i \le d^2, \quad x_{\min} = (1, 2, \ldots, d), \quad f_{\min} = -d(d+4)(d-1)/6.$$

2.2.3. Функции "плато"

Эти функции имеют плоские области или плато, что может замедлить работу алгоритмов оптимизации. Линейные методы и простой градиентный спуск часто оказываются здесь неэффективными, в то время как эвристические подходы, использующие случайную выборку или имитацию отжига, могут быть более эффективными.

McCormick Function Функция `McCormick` - это двумерная функция с относительно простой поверхностью, но невыпуклыми свойствами. Плато может ввести в заблуждение градиентные алгоритмы, но эвристические методы справляются неплохо.

$$f(x_1, x_2) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1, \tag{8}$$

$$-1.5 \leq x_1 \leq 4, -3 \leq x_2 \leq 4, \quad x_{\min} \approx (-0.54719, -1.54719), \quad f_{\min} \approx -1.9133.$$

Booth Function Функция `Booth` широко используется в оптимизации для тестирования алгоритмов с двумя переменными. Хотя функция имеет простую структуру, области плато могут замедлить сходимость для линейных или простых градиентных методо

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2, \tag{9}$$

$$-10 \leq x_1, x_2 \leq 10, \quad x_{\min} = (1, 3), \quad f_{\min} = 0.$$

2.2.4. Проблемы "долины"

Эти функции имеют узкие долины, в которых трудно найти глобальный минимум. Градиентные алгоритмы могут колебаться в пределах долины, прежде чем сходятся. Алгоритмы с адаптивным размером шага или гибридные метаэвристические методы могут работать лучше.

Rosenbrock Function The `Rosenbrock`, также известная как функция Банана, является классическим эталоном с изогнутой долиной, ведущей к глобальному минимуму. Градиентные методы часто оказываются неэффективными из-за плоскости вдоль оси долины, что требует применения продвинутых импульсных методов.

$$f(x) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right], \tag{10}$$

$$-5 \leq x_i \leq 10, \quad x_{\min} = (1, \ldots, 1), \quad f_{\min} = 0.$$

Six-Hump Camel Function The Six-Hump Camel function is a challenging two-dimensional test case with multiple local minima. It requires algorithms with global search capabilities, such as simulated annealing or genetic algorithms, to avoid getting stuck in local minima.

$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4, \tag{11}$$

$$-3 \leq x_1 \leq 3, -2 \leq x_2 \leq 2, \quad x_{\min} \approx (\pm 0.0898, \mp 0.7126), \quad f_{\min} \approx -1.0316.$$

2.2.5. Проблемы "хребтов и падений"

Функции этой категории характеризуются резкими изменениями градиента, что делает их труднопроходимыми для алгоритмов оптимизации. Метаэвристические методы, выполняющие широкую выборку, подходят лучше, чем традиционные градиентные подходы.

Easom Function Функция `Easom` имеет один резкий глобальный минимум, окруженный крутыми гребнями. Эта функция бросает вызов как градиентным, так и квазиградиентным алгоритмам, требуя стратегий глобального поиска для эффективного нахождения минимума.

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right), \tag{12}$$

$$-100 \leq x_1, x_2 \leq 100, \quad x_{\min} = (\pi, \pi), \quad f_{\min} = -1.$$

Michalewicz Function Функция `Michalewicz` - это мультимодальная проблема, зависящая от параметров d (размерность) и m (резкость минимумов). Крутые спады и узкие минимумы затрудняют работу большинства детерминированных методов, делая метаэвристические алгоритмы, такие как оптимизация муравьиной колонии или моделированный отжиг, более эффективными.ментальные методы.

$$f(x) = -\sum_{i=1}^{d} \sin(x_i) \left[\sin\left(\frac{ix_i^2}{\pi} \right) \right]^{2m}, \tag{13}$$

$$0 \leq x_i \leq \pi, \quad x_{\min} \text{ varies with } d, m.$$

3. Результаты

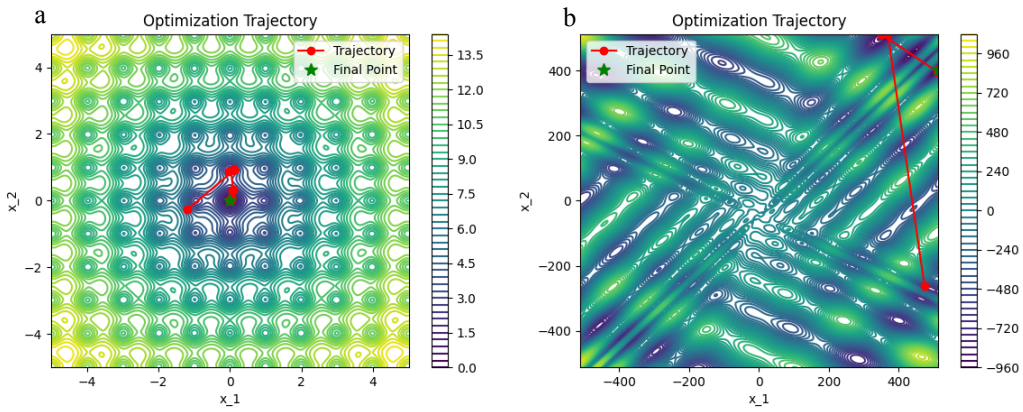

Рис. 1: Траектория поиска для проблем с множественным локальным минимумом. a - `Ackley`, b - `Eggholder`

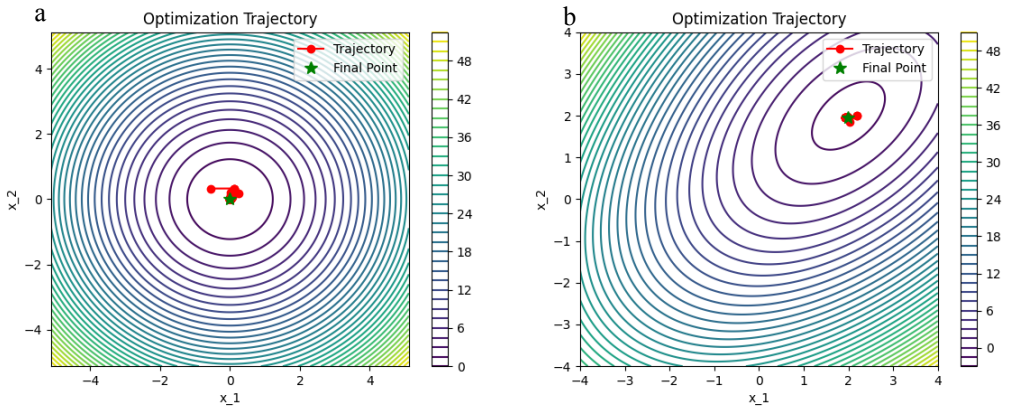

Рис. 2: Траектория поиска для проблем .... a - `Sphere`, b - `Trid`

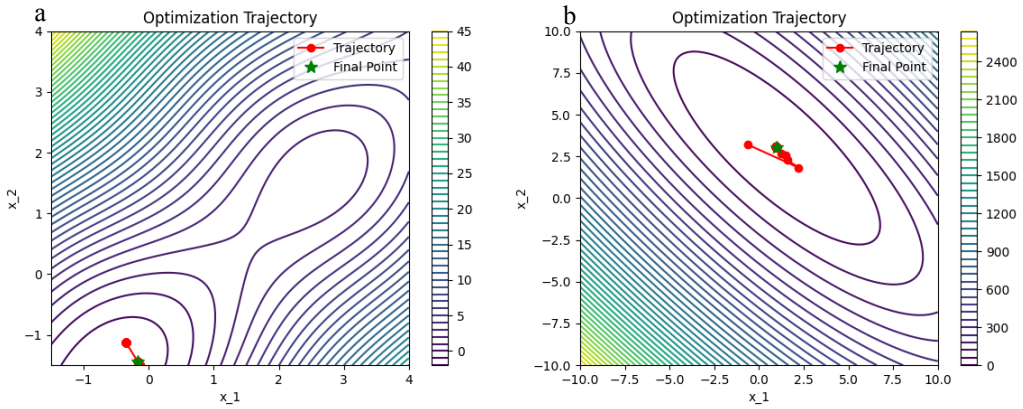

Рис. 3: Траектория поиска для проблем .... a - `McCormick`, b - `Booth`

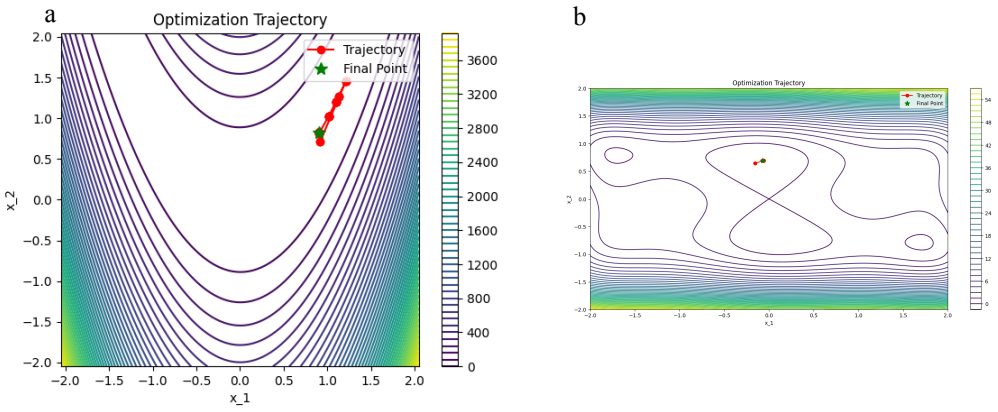

Рис. 4: Траектория поиска для проблем.... a - `Rosenbrock`, b - `Six-Hump Camel`

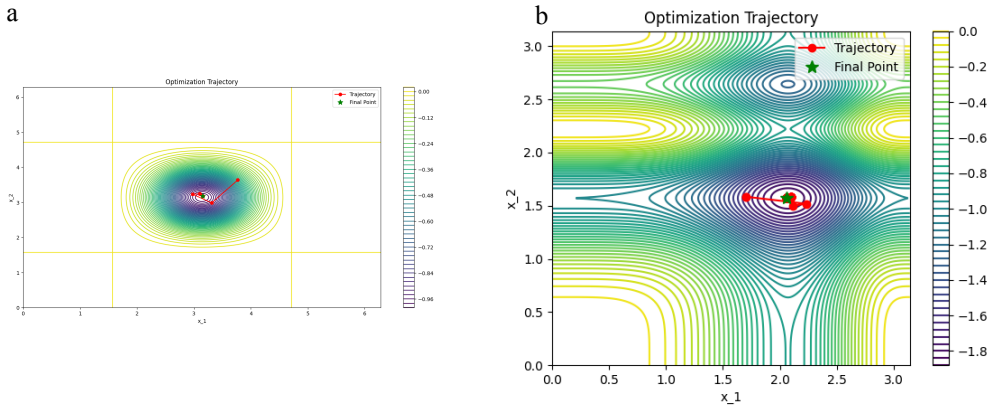

Рис. 5: Траектория поиска для проблем.... a - `Easom`, b - `Michalwicz`

4. Выводы

Для функции Ackley, характеризующейся множеством локальных минимумов, алгоритм эффективно показал способность избегать преждевременной сходимости. Быстрое исследование пространства с помощью γ-частиц позволяет охватить всю область поиска, в то время как α-частицы уточняют решение вблизи глобального минимума.

Функция Booth, будучи унимодальной и выпуклой, продемонстрировала быстрое достижение глобального минимума. Замедленное движение α-частиц помогает точному нахождению решения, в то время как β- и γ-частицы обеспечивают глобальный охват.

Резкий пик функции Easom и узкий глобальный минимум создают сложность для многих оптимизаторов. γ-частицы быстро находят область, где расположен минимум, а α-частицы успешно сужают поиск для достижения точного результата.

Сложный ландшафт функции Eggholder с крутыми гребнями и долинами демонстрирует способность оптимизатора работать в таких условиях. γ-частицы эффективно исследуют пространство, находя долины, а β-и α-частицы постепенно уточняют решения.

Гладкость и простая нелинейность функции McCormick позволяют алгоритму быстро сходиться. Баланс между различными типами частиц приводит к быстрому достижению минимума.

Обманчивый характер функции Michalewicz, включающий узкие долины и сложный рельеф, подчёркивает преимущества алгоритма. γ-частицы предотвращают застревание в локальных минимумах, в то время как α-частицы уточняют результаты.

Функция Rosenbrock с характерной "банановидной" долиной представляет вызов для многих оптимизаторов. β-частицы играют важную роль в навигации по долине, тогда как α-частицы обеспечивают точное достижение минимума.

Для функции с несколькими локальными минимумами, как Six-Hump Camel, алгоритм демонстрирует способность эффективно находить глобальный минимум благодаря сбалансированной работе γ, β и α-частиц.

Гладкая и выпуклая природа функции Sphere позволяет алгоритму быстро находить минимум. Сочетание различных частиц обеспечивает высокую скорость сходимости.

Функция Trid с уникальной кривизной и нелинейностью демонстрирует эффективность β-частиц в балансе между локальным и глобальным поиском, в то время как α-частицы уточняют решение.

1. **Исследование и уточнение:** Разделение ролей между γ, β и α-частицами позволяет эффективно балансировать между глобальным поиском и локальной оптимизацией.

2. **Адаптивность:** Алгоритм показал гибкость при работе с разнообразными тестовыми функциями, включая мультиэкстремальные и унимодальные.

3. **Сходимость:** Взаимодействие частиц обеспечивает надёжное достижение глобального минимума без застревания в локальных экстремумах.

Таким образом, результаты тестирования подтверждают эффективность и универсальность оптимизатора CDO.