ЗАДАНИЕ 5
Метод роя частиц

Построение модели • Рассмотрим реализацию метода роя частиц на примере решения задачи минимизации двумерной функции действительного аргумента.

A Создаем новую модель. Устанавливаем ее размеры pxcor-max и pycor-max равными 100, размер патча — равным 2 пикселям.

B Добавляем к интерфейсу модели выпадающий список target-function, который будет задавать тип целевой функции, подлежащей минимизации. Для начала ограничимся двумя стандартными тестовыми функциями — сферической (простейшая унимодальная функция, геометрически — параболоид вращения, рис. 5.1):

$$f(x,y) = x^2 + y^2 \tag{5.1}$$

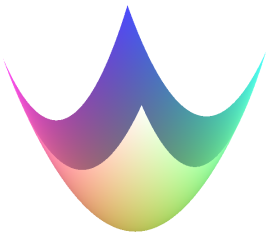

РИС. 5.1 Сферическая функция

и функцией Растригина (мультимодальная функция, рис. 5.2):

$$f(x,y) = x^2 + y^2 + 10(2 - \cos(2\pi x) - \cos(2\pi y)). \tag{5.2}$$

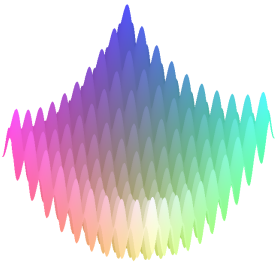

РИС. 5.2 Функция Растригина

Обе функции имеют глобальный минимум в начале координат. Включим в список соответствующие два элемента: `"sphere"` и `"rastrigin"`.

C Переходим во вкладку с кодом и создаем атрибут для патчей `val` — значение целевой функции в этом патче, будем вычислять его один раз при инициализации модели.

D Создаем функцию `eval-target` для вычисления выбранной пользователем целевой функции в

точке с координатами x и y. Перед применением формул (5.1) и (5.2) выполняем нормировку координат x и y, поделив их на 10. Таким образом, новые координаты точки будут лежать в диапазоне от −10 до 10. Для функции Растригина в каждом косинусе замените 2π (радианы) на 360 (градусы).

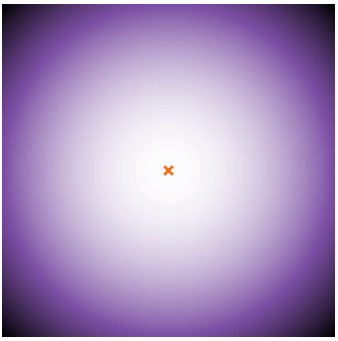

РИС. 5.3 Вид модели для сферической функции

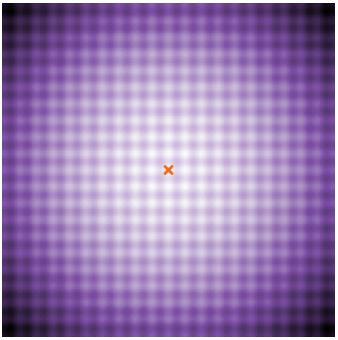

РИС. 5.4 Вид модели для функции Растригина

[1] Темный оттенок (фактически черный цвет) будет соответствовать первому указанному пределу, светлый (белый цвет) — второму пределу.

E Добавляем кнопку setup и соответствующую ей процедуру, в которой очищаем мир, вызываем процедуру setup-landscape для формирования и визуализации целевой функции и сбрасываем таймер.

F Пишем код процедуры setup-landscape. Сначала просим все патчи вычислить и запомнить в атрибуте val значение целевой функции в точке со своими координатами:

```
set val eval-target pxcor pycor.
```

После этого находим минимальное и максимальное значения minv и maxv атрибута val среди всех патчей. Третьим шагом раскрашиваем патчи в зависимости от значения атрибута val. Для этого можно использовать встроенную функцию scale-color, которая принимает четыре параметра:

- базовый цвет (violet);
- значение, по которому вычисляется оттенок базового цвета (атрибут val);
- два предела изменения этого значения (maxv и minv)[1].

Полная команда установки цвета патча выглядит следующим образом:

```
set pcolor scale-color violet val maxv minv.
```

Наконец, помещаем в точку глобального минимума, т.е. в начало координат, одну черепаху, которая будет указывать положение нашей цели. Определяем визуальные характеристики цели: цвет — оранжевый, размер — 8, форма — "x" (крестик).

G Проверяем работу кнопки setup для обеих целевых функций (рис. 5.3 и 5.4).

H Переходим к созданию роя частиц. Добавляем к интерфейсу модели слайдер particles-number с диапазоном значений от 2 до 30, шагом 1 и значением по умолчанию 10.

I Объявляем новый вид черепах `particles`. Добавляем атрибуты для частиц: `vx` и `vy` — компоненты скорости, `p-val` — лучшее найденное значение целевой функции для данной частицы, `p-x` и `p-y` — координаты лучшей точки траектории данной частицы.

J Создаем три глобальные переменные: `g-val` — лучшее найденное роем значение целевой функции, `g-x` и `g-y` — координаты соответствующей точки.

K В процедуре `setup` после формирования целевой функции создаем рой, состоящий из заданного числа (`particles-number`) частиц. Каждую частицу помещаем в случайную точку и определяем ее атрибуты:

- форма — круг;
- размер — 4;
- компоненты скорости `vx` и `vy` полагаем равными компонентам текущей скорости черепахи `dx` и `dy`;
- значение `p-val` — значению `val` патча, в котором располагается черепаха: `set p-val val`;
- координаты `p-x` и `p-y` — текущим координатам черепахи `xcor` и `ycor`.

L Добавляем кнопку `go` и соответствующую процедуру, в которой пока выполняем простое движение частиц без модификации их скорости.

```
ask particles [
  facexy xcor + vx ycor + vy
  fd sqrt (vx ^ 2 + vy ^ 2)
]
```

Команда `facexy` разворачивает частицу в заданном направлении (в нашем случае в направлении ее текущей скорости), команда `fd` передвигает частицу вперед на величину ее скорости. Обновляем таймер и проверяем работу модели (кнопка `go`).

M Добавляем к интерфейсу модели переключатель `trace?` для отслеживания траекторий движения частиц (рис. 5.5). В начале процедуры `go` проверяем значение этого переключателя: если он выключен, то удаляем все нарисованные траектории, после этого просим все частицы опустить или поднять перо в зависимости от выбранного режима.

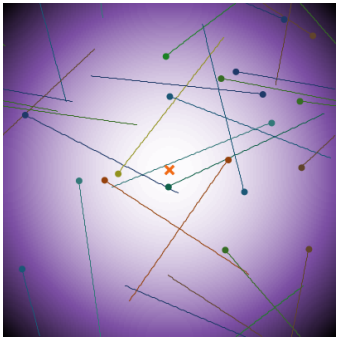

РИС. 5.5 Простое движение частиц с отслеживанием траекторий

```
if not trace? [clear-drawing]
ask particles [
  ifelse trace? [pen-down] [pen-up]
]
```

N В процедуре `go` просим каждую частицу сравнить значение целевой функции в текущем патче с запомненным в атрибуте `p-val` значением. Если текущее значение целевой функции меньше запомненного, то обновляем значения атрибутов `p-val`, `p-x` и `p-y`.

O Вычисляем в процедуре `go` глобальное лучшее значение целевой функции `g-val` и его положение `p-x` и `p-y` — это лучшее значение атрибута `p-val` среди всех частиц:

```
ask min-one-of particles [p-val] [
  set g-val p-val
  set g-x p-x
  set g-y p-y
]
```

P Пишем код обновления скорости, которое выполняется перед движением частицы в два шага — локальное и глобальное обновление. Добавляем к интерфейсу два слайдера alpha-max и beta-max (диапазон от 0 до 0.1, шаг 0.01, значение по умолчанию 0.05), которые определяют максимальные значения коэффициентов α и β в (5.3).

$$v_k \leftarrow v_k + \alpha \cdot (p_k - x_k) + \beta \cdot (g - x_k). \qquad (5.3)$$

Q На первом шаге к текущей скорости частицы прибавляем вектор, направленный на лучшую точку траектории данной частицы, если частица находится не в этой точке. Для этого находим направление `h` на лучшую точку с помощью команды `set h towardsxy p-x p-y` и расстояние `d` до этой точки с помощью команды `set d distancexy p-x p-y`. После этого вычисляем случайное значение коэффициента `alpha` и обновляем компоненты скорости `vx` и `vy`:

```
let alpha random-float alpha-max
set vx vx + alpha * d * sin h
set vy vy + alpha * d * cos h
```

R На втором шаге обновления скорости повторяем действия с первого шага, но уже применительно к точке (`g-x`, `g-y`) с лучшим значением целевой функции по всему рою. При этом заменяем па-

раметр alpha на beta. Проверяем работу модели (рис. 5.6).

S Если дать модели поработать некоторое время, то можно заметить, что некоторые частицы начинают постепенно разгоняться, что негативно сказывается на процессе решения задачи. Подавить этот эффект можно двумя способами. Во-первых, можно просто ограничить максимальную скорость частиц некоторым значением v-max. Для этого добавьте соответствующий элемент ввода, установите его значение равным 2. Перед перемещением частицы проверьте величину ее скорости, если она больше v-max, то сделайте ее равной v-max.

T Второй способ состоит в модификации формулы (5.3), надо перед первым слагаемым (инерция частицы) поставить множитель $\gamma < 1$, который будет представлять сопротивление среды. Добавьте соответствующий элемент ввода к интерфейсу и модифицируйте код процедуры go. Пример работы метода для функции Растригина с ограничением скорости и сопротивлением $\gamma = 0.999$ приведен на рис. 5.7.

U Добавьте к интерфейсу график, показывающий сходимость метода. Одна кривая должна показывать лучшее текущее значение целевой функции в рое, вторая — ее среднее значение (рис. 5.8). Проверьте работу метода для обеих целевых функций в разных режимах (для разных значений параметров модели). Окончательный интерфейс модели показан на рис. 5.9.

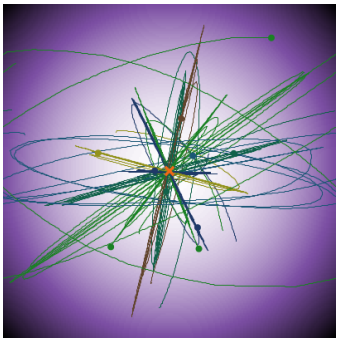

РИС. 5.6 Траектории движения частиц в методе роя частиц для сферической целевой функции

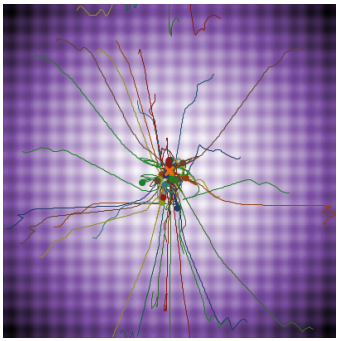

РИС. 5.7 Траектории движения частиц при ограничении скорости

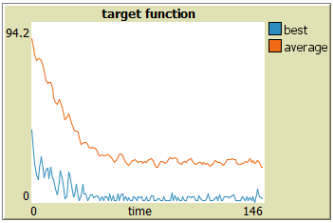

РИС. 5.8 Сходимость метода роя частиц

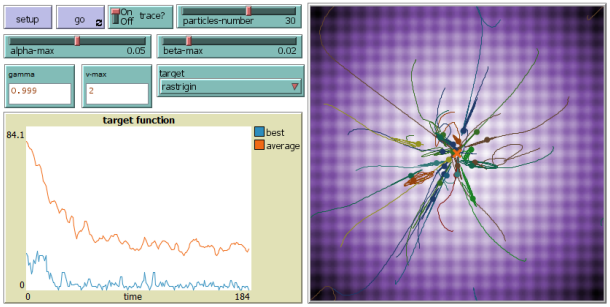

РИС. 5.9 Окончательный интерфейс модели

УПРАЖНЕНИЯ

1 Реализуйте модифицированную модель Рейнолдса, в которой птицы выполняют целенаправленный поиск кормушки.

2 Включите в рассмотренную в настоящей главе модель поддержку других целевых функций[2], в том числе:

- функции Шаффера;
- функции Розенброка, глобальный минимум в точке $(1, 1)$ на дне оврага (рис. 5.10):

$$f(x, y) = 50(y - x^2)^2 + (x - 1)^2.$$

- функции Акли, глобальный минимум в начале координат (рис. 5.11):

$$f(x, y) = 20\left(1 - e^{-\frac{\sqrt{x^2+y^2}}{2}}\right) + e - e^{\frac{\cos 2\pi x + \cos 2\pi y}{2}}.$$

Т.к. у разных целевых функций (например, у функции Розенброка) глобальный минимум находится не обязательно в начале координат, измените код процедуры `setup-landscape` так, чтобы положение этого минимума определялось автоматически.

3 Включите в модель случайную целевую функцию (рис. 5.12), сгенерированную по следующему алгоритму. Каждому патчу приписывается случайное значение атрибута `val` из диапазона от 0 до 1. После этого выполняется сглаживание функции с помощью многократного выполнения встроенной команды `diffuse`:

```
repeat 50 [diffuse val 0.5].
```

4 Рассмотрите вариацию метода роя частиц, в которой величина скорости частиц фиксирована, а ее направление выбирается по формуле (5.3). Включите в эту вариацию схему имитации отжига для скорости частиц — в начальный момент времени скорость устанавливается высокой, а затем постепенно снижается до нуля. Сравните сходимость этого метода с оригинальной версией.

5 Если запустить модель для функции Растригина с небольшим количеством частиц (до десяти), то можно увидеть характерный эффект (рис. 5.13, слева), когда частицы начинают двигаться по сильно вытянутым «орбитам». Этот эффект является непосредственным

[2] J. Dieterich, B. Hartke, *Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization*, Applied Mathematics, Vol. 3 No. 10A, 2012, p. 1552–1564.

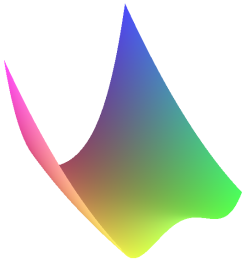

РИС. 5.10 Функция Розенброка

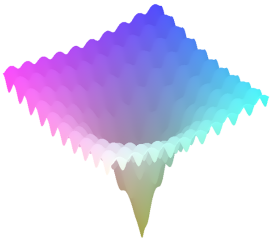

РИС. 5.11 Функция Акли

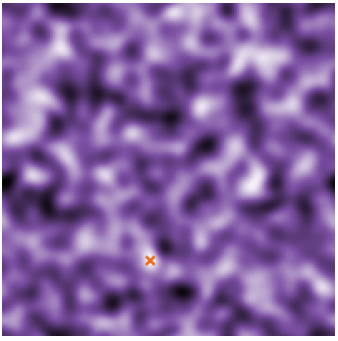

РИС. 5.12 Случайно сгенерированная целевая функция

следствием правила (5.3): если точка текущего положения частицы, ее лучшая точка и лучшая точка роя оказались лежащими на одной прямой, то частица не сможет сойти самостоятельно с этой прямой. Один из способов исключения данного эффекта, который негативно влияет на сходимость метода, заключается в выборе случайных весовых коэффициентов α и β *отдельно* для каждой компоненты (координаты) скорости (рис. 5.13, справа). Реализуйте эту схему и проанализируйте, как она влияет на сходимость метода.

РИС. 5.13 Влияние способа выбора параметров α и β на характер траекторий частиц

6 Использование глобального лучшего решения g в методе роя частиц ускоряет его сходимость. С другой стороны, при минимизации мультимодальных функций учет глобального вектора g обычно заводит метод в какой-нибудь локальный минимум целевой функции, из которых метод не способен выбраться самостоятельно. Эта проблема является общей практически для всех популяционных алгоритмов оптимизации и называется проблемой потери разнообразия: в результате работы алгоритма все особи популяции сосредоточиваются в окрестности одной точки (локального экстремума), т.е. оказываются почти неотличимыми друг от друга, что и приводит фактически к стагнации процесса решения. Для метода роя частиц одним из способов отложить эту проблему (за счет замедления сходимости) является введение вместо единого глобального вектора g отдельных, привязанных к частицам, локальных векторов g_k. Перед обновлением скорости каждая частица опрашивает своих соседей по пространству решений, находящихся в ее окрестности размера `vision`, выбирает частицу с наилучшим значением `p-val` и использует ее атрибуты `p-x` и `p-y` в качестве значений `g-x` и `g-y` для обновления своей скорости. Реализуйте описанную схему и проверьте ее работу на мультимодальных целевых функциях. На рис. 5.14 показаны примеры работы этой вариации метода роя частиц для нескольких значений параметра `vision` для функции Растригина. Хорошо видно, что при малых значениях этого параметра (слева вверху) частицы «обживают» сразу несколько локальных минимумов целевой функции, а при увеличении `vision` частицы начинают сосредоточиваться в окрестности глобального минимума.

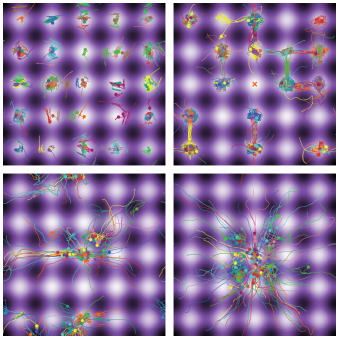

РИС. 5.14 Траектории частиц в локальном варианте метода роя частиц для разных значений параметра `vision`

7 Другой, так называемый *социальный* способ внесения локальности в метод роя частиц заключается в том, что частицы выбирают лучшего соседа не по пространству решений, а по некоторому отношению, определенному на множестве частиц. В простейшем случае соседями частицы с номером k объявляются частицы, чьи

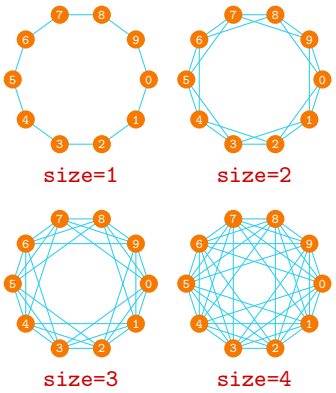

РИС. 5.15 Примеры отношения соседства на множестве частиц

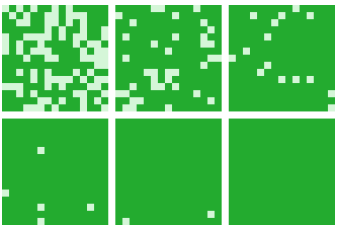

РИС. 5.16 Пример работы метода роя частиц в задаче OneMax

номера отличаются от k не более чем на заданную величину `size` (с учетом зацикливания, т.е. соседом последней частицы будет первая, и наоборот). Это отношение можно наглядно представить в виде графа (рис. 5.15), своего рода социальной сети роя частиц. Реализуйте и протестируйте этот подход.

8 Примените метод роя частиц к задаче одномерной минимизации, взяв за основу модель для алгоритма имитации отжига, рассмотренную нами в предыдущей главе. Сравните скорость сходимости этих двух методов для разных целевых функций.

9 Отличительной особенностью метода роя частиц от многих других методов является то, что для него необходимо уметь вычислять только значение оптимизируемой функции, но не ее градиент. Т.е. функция, подлежащая оптимизации, не обязана быть дифференцируемой, более того, она может быть разрывной, зашумленной и т.п. С другой стороны, в силу того, что метод оперирует понятием скорости частиц, необходимым условием его применимости является *непрерывность* области определения функции. В частности, это означает, что данный метод неприменим напрямую к задачам дискретной оптимизации. Проблемой является то, что в дискретном пространстве решений понятие скорости частицы полностью теряет свой смысл. Один из вариантов решения этой проблемы для случая задач с двоичным кодированием заключается в том, что каждая компонента скорости частицы, ограниченная интервалом $[0, 1]$, трактуется как *вероятность* изменения соответствующей координаты этой частицы: чем больше скорость, тем чаще меняется (инвертируется) данная координата. Изменение скорости в направлении заданного решения (второе и третье слагаемые в (5.3)) выполняется по следующему правилу: если биты двух решений совпадают, то скорость уменьшается, если различаются — то увеличивается. Постройте модель для такого метода роя частиц и протестируйте ее на задаче *OneMax*: требуется найти двоичную последовательность заданной длины, в которой количество единичных цифр было бы максимальным. Для визуализации двоичных последовательностей используйте патчи модели (рис. 5.16).

10 Если пространство решений в задаче дискретной оптимизации представляет собой множество перестановок из n элементов (например, в задаче коммивояжера), то для адаптации метода роя частиц можно использовать следующий прием. Рой частиц «живет» в обычном непрерывном n-мерном пространстве. Коор-

динаты k-й частицы перед вычислением целевой функции преобразуются в перестановку по следующему алгоритму: упорядочиваем вектор координат данной частицы $(x_1, x_2, \ldots, x_n)$ по возрастанию, номера индексов в упорядоченной последовательности и образуют искомую перестановку. Например, следующий вектор[3]

$$x = (2.1, 7.9, 11.0, 3.3, 1.8, 4.2)$$

будет преобразован в перестановку (проверьте!)

$$\pi = (5, 1, 4, 6, 2, 3).$$

Разработайте модель для данной вариации метода роя частиц применительно к решению задачи коммивояжера. На рис. 5.18 приведен пример работы такого алгоритма к задаче коммивояжера, в которой расстояние между городами вычисляется не в евклидовой метрике

$$\rho_e(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2},$$

а в так называемой манхэттенской метрике[4]:

$$\rho_m(A, B) = |x_A - x_B| + |y_A - y_B|.$$

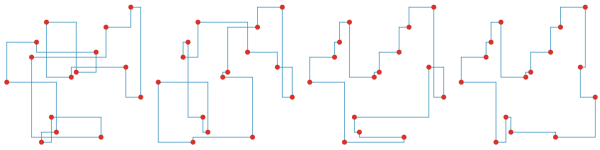

РИС. 5.18 Сходимость метода роя частиц при решении задачи коммивояжера в манхэттенской метрике

11 Относительно недавняя вариация метода роя частиц, превосходящая его в некоторых задачах, — алгоритм CSO (Competetive Swarm Optimizer[5]). В этом алгоритме на каждом шаге частицы делятся случайным образом на пары, между частицами в каждой паре происходит *турнир*, т.е. сравнение целевых функций (рис. 5.19). Проигравшая турнир частица, имеющая худшее значение целевой функции, выполняет шаг *обучения* — сначала она меняет свою скорость v по следующей формуле:

$$v \leftarrow v + \alpha \cdot (x_{win} - x) + \beta \cdot (\bar{x} - x), \qquad (5.4)$$

где x — ее текущее положение, x_{win} — текущее положение частицы, выигравшей турнир, $\bar{x}$ — координаты центра масс роя. После этого проигравшая частица меняет свое положение по стандартной схеме:

$$x \leftarrow x + v.$$

[3] Нумерация элементов вектора (т.е. координат частицы) начинается с единицы.

[4] Другое название этой метрики — расстояние городских кварталов (рис. 5.17).

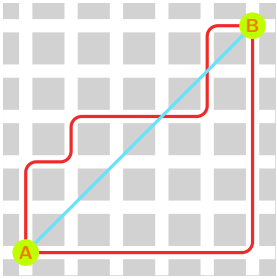

РИС. 5.17 Сравнение евклидовой (голубая линия) и манхэттенской метрик (красные линии)

[5] R. Cheng, Y. Jin, *A Competitive Swarm Optimizer for Large Scale Optimization*, in IEEE Transactions on Cybernetics, Vol. 45, No. 2, Feb. 2015, p. 191–204.

Частица-победитель своего состояния не меняет. Реализуйте этот метод, взяв за основу разработанную в главе модель. Выполните сравнение эффективности методов PSO и CSO на тестовых задачах.

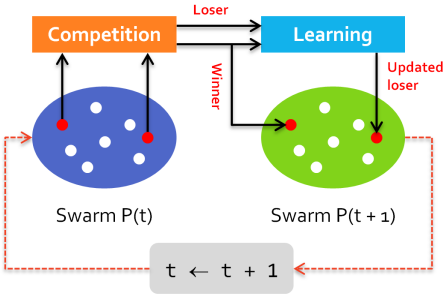

РИС. 5.19 Схема работы метода CSO