ЗАДАНИЕ 3

Метод имитации отжига

Построение модели • Метод имитации отжига на практике целесообразно применять к сложным задачам (дискретная оптимизация, оптимизация функций высокой размерности), для решения которых нет более эффективных алгоритмов. Мы, однако, в иллюстративных и ознакомительных целях рассмотрим задачу минимизации одномерной непрерывной функции $f(x)$, $x \in \mathbb{R}$. Некоторые более сложные задачи оптимизации будут рассмотрены в упражнениях.

A Создаем новую модель и настраиваем ее размеры: значение `max-pxcor` делаем равным 24, значение `max-pycor` — равным 12.

B Добавляем кнопку `setup`, пишем код процедуры `setup`, в котором очищаем мир, раскрашиваем патчи в белый цвет, сбрасываем таймер.

C Модель будет работать с предопределенными функциями $f(x)$, для выбора которых добавим к интерфейсу модели выпадающий список с именем `target`. Для начала включим в этот список две опции:

- `"unimodal"` — функция с одним минимумом;
- `"multimodal"` — функция с несколькими минимумами (из которых только один глобальный).

К коду модели добавим функцию `func [x]`, которая вычисляет значение выбранной функции в точке `x`. Будем предполагать, что все функции рассматриваются на интервале от 0 до 1, и в таком же интервале лежат все их значения. В качестве

унимодальной функции будем использовать квадратичную функцию вида

$$f(x) = (2x - 1)^2 \tag{3.1}$$

с вершиной в $x = 1/2$. В качестве мультимодальной функции выберем функцию следующего вида[1]:

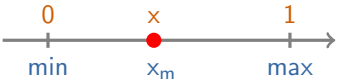
$$f(x) = \frac{2 + \cos 720x + \cos 2000x}{4}. \tag{3.2}$$

[1] Еще раз напомним, что все тригонометрические функции в NetLogo работают с градусами, поэтому такие «большие» коэффициенты в косинусах.

D Наши функции будут определены на интервале $[0, 1]$ с областью значений из такого же интервала. Т.е. все должно происходить в единичном квадрате с левым нижним углом в начале координат. Любая модель в NetLogo использует свою систему координат, привязанную к патчам. Например, горизонтальные координаты в модели изменяются от `min-pxcor - 0.5` до `max-pxcor + 0.5`[2]. Для этого нам потребуются две функции (для каждой из двух координат), которые преобразуют координаты точки из единичного квадрата в координаты модели, причем $x = 0$ должно отображаться в `min-pxcor`, а $x = 1$ — в `max-pxcor`[3] (аналогично для y-координаты). Формула преобразования x-координаты в координату x_m модели определяется простой пропорцией (рис. 3.1, значения min и max на рисунке соответствуют `min-pxcor` и `max-pxcor`) и записывается в NetLogo следующей функцией.

[2] Все патчи центрированы относительно своих целочисленных координат, а размер патча всегда равен 1.

[3] Таким образом, в нашей модели будут оставаться узкие (шириной в 0.5 патча) поля.



$$\frac{x_m - \min}{x - 0} = \frac{\max - \min}{1 - 0}$$

РИС. 3.1 Схема преобразования координат

```
to-report get-xcor [x]
  report min-pxcor + x * (max-pxcor - min-pxcor)
end
```

Создаем аналогичную функцию `get-ycor` для преобразования заданной y-координаты.

E Добавим к коду процедуры `setup` (перед сбросом таймера) вызов процедуры `draw-target`, которая будет строить график целевой функции. Создаем процедуру `draw-target`. Идея рисования графика заключается в создании одной черепахи, которая будет двигаться по траектории, определяемой функцией $y = f(x)$, и рисовать свой след. Процесс рисования начинается с точки `(0, func 0)`. Затем делается 200 шагов по оси абсцисс (от 0 до 1 с шагом $1/200$), на каждом шаге вычисляется координата `x`, после этого черепаха перемещается в новую

точку с координатами `(x, func x)`. После построения графика черепаха удаляется из модели.

```
to draw-target
  create-turtles 1 [
    set color black
    let i 0
    setxy get-xcor 0 get-ycor func 0
    pendown
    repeat 200 [
      set i i + 1
      let x i / 200
      setxy get-xcor x get-ycor func x
    ]
    die
  ]
end
```

F Проверяем процесс рисования графика целевой функции (т.е. работу кнопки setup). На рис. 3.2 показаны графики двух функций, которые должна строить наша модель.

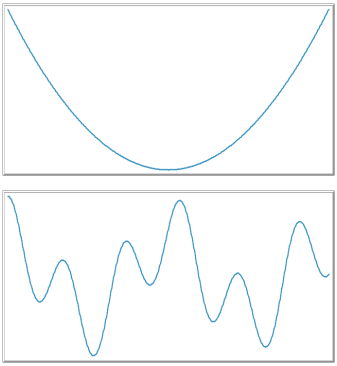


РИС. 3.2 Унимодальная и мультимодальная целевые функции в модели имитации отжига

G В нашей модели мы реализуем вариант алгоритма имитации отжига сразу для нескольких частиц (черепах, в терминах NetLogo). Это позволит нам строить распределения частиц по области решения более простым и быстрым способом, по сравнению со случаем одной частицы. С этой целью добавим к интерфейсу слайдер turtles-number с диапазоном от 1 до 500.

H Каждая черепаха соответствует одному пробному решению задачи оптимизации и должна хранить в своих атрибутах это решение. В нашем случае решением является единственное число x из интервала от 0 до 1. Припишем черепахам атрибут `my-x`, в котором и будет храниться пробное решение[4].

I В процедуре `setup` создадим соответствующее число (`turtles-number`) черепах, для каждой черепахи вызовем процедуру `setup-turtle` для настройки ее атрибутов.

J Пишем код процедуры `setup-turtle`, в которой устанавливаем круглую форму черепах, задаем их размер — 0.5, выбираем случайное пробное решение $x \in [0, 1]$: `set my-x random-float 1`. Наконец, переносим черепаху в точку $(x, f(x))$:

```
setxy get-xcor my-x get-ycor func my-x.
```

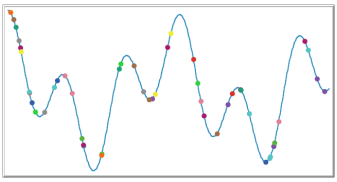


РИС. 3.3 Начальный вид модели для мультимодальной целевой функции

[4] Новые атрибуты добавляются к любому виду агентов с помощью команд типа `turtles-own`, первое слово в имени команды — вид агентов во множественном числе. Команды такого типа должны быть записаны *в начале кода*. В нашем случае мы первой строкой кода пишем команду:

```
turtles-own [my-x].
```

K Проверяем работу кнопки setup. Внешний вид модели для мультимодальной функции приведен на рис. 3.3.

L Переходим к реализации алгоритма Метрополиса. Для этого добавим к интерфейсу модели следующие элементы:

- поле ввода theta0 (тип number) — начальная температура в модели;
- поле ввода vel (тип number) — величина максимального изменения пробного решения;
- слайдер theta с диапазоном от 0.0001 до 10 и шагом 0.1 — текущая температура в модели, с помощью этого элемента можно будет вручную управлять отжигом;
- кнопку go.

M В процедуре setup инициализируем температуру theta значением theta0.

N Создаем процедуру go, в которой для каждой черепахи вызываем процедуру move-turtle для ее случайного перемещения в новую точку. Не забываем обновить таймер.

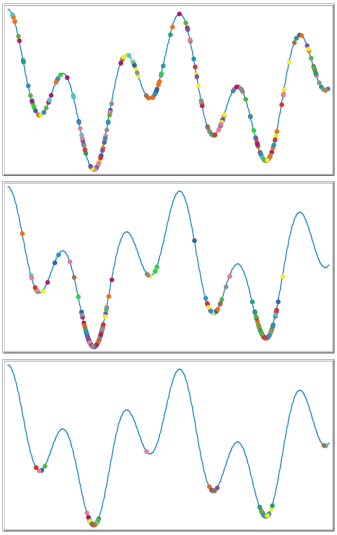


РИС. 3.4 Поведение модели для трех значений температуры (сверху вниз: высокая, средняя, низкая)

O Пишем код процедуры move-turtle, реализующей алгоритм Метрополиса.

```
to move-turtle
  let new-x my-x + vel * (1 - random-float 2)
  if new-x < 0 or new-x > 1 [stop]
  let old-y func my-x
  let new-y func new-x
  let d (new-y - old-y) / theta
  if d < 0 or random-float 1 < exp (- d) [
    set my-x new-x
    setxy get-xcor my-x get-ycor new-y
  ]
end
```

Сначала вычисляем новое пробное положение частицы new-x, если оно выходит за пределы интервала $[0, 1]$, то отвергаем это некорректное решение. Затем вычисляем текущее и пробное значения целевой функции и находим их разность d, деленную на текущую температуру theta. После этого применяем правило Метрополиса: меняем текущее решение на пробное и перевычисляем новое положение частицы, если новое решение лучше текущего или с вероятностью e^{-d} в противном случае.

P Проверяем работу модели, вручную управляя ее температурным режимом (с помощью слайдера theta). Если сделать температуру высокой, то частицы распределятся более-менее равномерно по всей области решения. Если сделать температуру низкой, то частицы должны собраться в локальных минимумах целевой функции (рис. 3.4).

Q Включим в модель процедуру отжига. Для этого к интерфейсу добавим переключатель cool?, отвечающий за включение и выключение процесса отжига, а также поле ввода q для управления скоростью отжига. В процедуре go добавим команду, которая уменьшает температуру при включенном режиме отжига:

```
if cool? [set theta theta * q].
```

R Добавим к интерфейсу график сходимости метода, который должен содержать две кривые, соответствующие лучшему и среднему значениям целевой функции среди всех частиц на каждой итерации алгоритма (рис. 3.5). Причем для большей информативности в графике сходимости лучше использовать логарифмический[5] масштаб по вертикальной оси. Добавим к коду две функции, которые будут вычислять указанные величины. Функция best-target находит лучшее (т.е. минимальное) значение с помощью команды min, принимающей на вход список значений.

```
to-report best-target
  let y min [func my-x] of turtles
  report log y 10
end
```

Функция average-target должна находить среднее значение, вычисление которого удобно выполнять с помощью команды reduce, первым аргументом которой идет операция (в нашем случае +), вторым — список значений. Найденную сумму делим на количество частиц (черепах).

```
to-report average-target
  let s reduce + [func my-x] of turtles
  let y s / count turtles
  report log y 10
end
```

В параметрах графика установите нулевое значение Y max[6], задайте цвет обеих кривых и определите команды их прорисовки. Например, для кри-

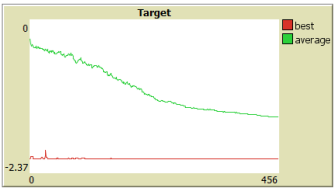


РИС. 3.5 График сходимости метода имитации отжига

[5] По основанию 10, чтобы легко можно было оценить порядок значения целевой функции.

[6] Нуль — это десятичный логарифм от максимального значения целевой функции в нашей модели.

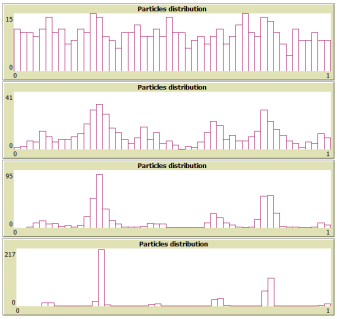


РИС. 3.6 Изменение распределения частиц в процессе отжига

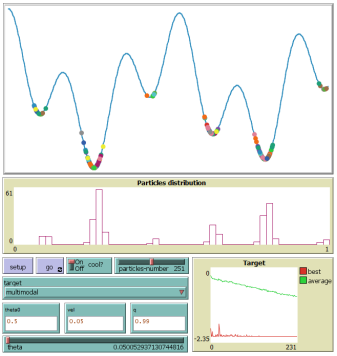


РИС. 3.7 Окончательный интерфейс модели

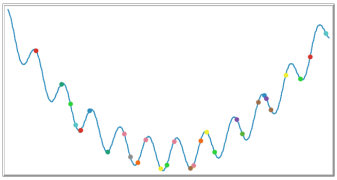


РИС. 3.8 Функция Растригина — один из стандартных тестов для алгоритмов непрерывной оптимизации

вой, соответствующей лучшему среди всех частиц значению целевой функции, используйте команду `plot best-target`.

S Последним пунктом в построении модели будет построение гистограммы распределения частиц по области поиска, т.е. по интервалу $[0, 1]$. Добавляем к интерфейсу график, располагая его непосредственно под самой моделью. Устанавливаем значение X max равным 1. В поле Plot setup commands определяем число столбиков гистограммы:

```
set-histogram-num-bars 50
```

В поле Pen update commands задаем команду рисования `histogram [my-x] of turtles`. В расширенных настройках кривой устанавливаем режим показа Bar. На рис. 3.6 показано, как меняется распределение частиц по области поиска в процессе отжига для мультимодальной целевой функции.

T Тестируем работу модели в разных режимах. Окончательный вид модели показан на рис. 3.7.

УПРАЖНЕНИЯ

1 Добавьте в модель возможность автоматического увеличения температуры в системе по закону

$$\theta_{k+1} = \theta_k / q.$$

Для управления режимом нагрева используйте отдельный переключатель `heat?`. Протестируйте этот режим для мультимодальной целевой функции, при условии что в начальный момент времени все частицы помещаются в одну и ту же точку с координатой $x = 0.1$, являющуюся локальным минимумом целевой функции. Проследите, как с увеличением температуры частицы начинают покидать это локальное решение, перемещаясь прежде всего в область глобального минимума.

2 Включите в модель целевую функцию Растригина

$$f(x) = 0.8(2x - 1)^2 + 0.1 + 0.1 \cos 4100x$$

с глобальным минимумом в точке $x = 0.5$ (рис. 3.8). Исследуйте, как параметр `vel` влияет на скорость сходимости алгоритма при минимизации этой функции.

3 Адаптируйте модель, чтобы она могла работать с функциями, области определения и значений которых не ограничены интервалом $[0, 1]$, а могут быть произвольными.

4 Предоставьте пользователю возможность задавать целевую функцию через интерфейс модели. Для этого надо вместо выпадающего списка со встроенными в модель функциями использовать элемент ввода типа string (пусть имя связанной с этим элементом переменной останется прежним — target). Теперь надо переписать код процедуры func:

```
to-report func [x]
  report runresult target
end
```

5 Существует несколько разных схем отжига, определяющих зависимость температуры θ от номера итерации $k > 0$ (рис. 3.9). Например, в схеме Больцмана температура убывает по закону

$$\theta(k) = \frac{\theta_0}{\ln(1+k)}.$$

В более быстрой схеме Коши применяется следующая формула:

$$\theta(k) = \frac{\theta_0}{k}.$$

Еще более быстрой схемой, в которой отжиг завершается за конечное число итераций, является линейная схема отжига:

$$\theta(k) = \max(\theta_{\min}, \theta_0 - k\triangle\theta).$$

Включите эти схемы (вместе с уже реализованной экспоненциальной схемой) в модель и предоставьте пользователю возможность их выбора.

6 Включите в модель режим супербыстрого отжига, когда температура θ сразу же полагается равной нулю. В таком режиме система может двигаться только строго вниз по целевой функции (энергии), т.е. метод имитации отжига превращается в вариант алгоритма безусловного спуска. Особенностью такого рода алгоритмов является то, что они находят только ближайший к начальной точке локальный минимум целевой функции, правда, делают это очень быстро.

7 Добавьте к интерфейсу модели график, показывающий, как распределяются частицы по величине целевой функции (энергии). Сравните с теоретическим распределением Больцмана:

$$f(x) = \frac{1}{Z} e^{-E(x)/k\theta}, \qquad (3.3)$$

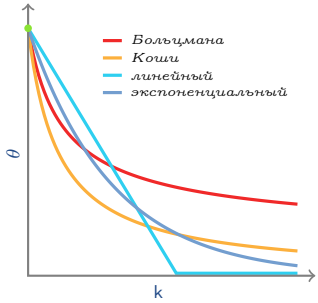


РИС. 3.9 Четыре режима охлаждения в методе имитации отжига

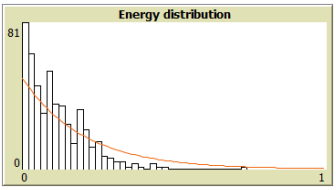


РИС. 3.10 Распределение частиц по величине целевой функции (Растригина) в сравнении с распределением Больцмана

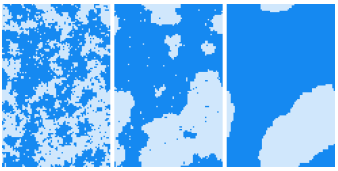


РИС. 3.11 Процесс отжига в модели Изинга с независимым изменением состояний частиц

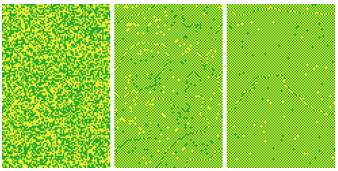


РИС. 3.12 Антиферромагнитный вариант модели Изинга с искусственным отжигом (слева направо: исходная случайная конфигурация, быстрый отжиг, медленный отжиг)

где k — постоянная Больцмана, Z — нормирующий множитель. Для этого необходимо экспериментально подобрать правильное значение постоянной Больцмана k. Например, на рис. 3.10 показано распределение энергии для функции Растригина (гистограмма) и распределение Больцмана с $k = 10$ (кривая).

8 Реализуйте модель Изинга. Каждый патч в модели соответствует одной частице, имеющей ровно четырех соседей. Каждой частице приписано состояние x, которое может принимать только два значения ± 1. Частицы взаимодействуют только с соседними частицами, энергия попарного взаимодействия определяется произведением состояний, взятым со знаком минус: $E(x_i, x_j) = -x_i x_j$. Полная энергия системы равна сумме энергий всех попарных взаимодействий. Рассмотрите два варианта выбора нового состояния в алгоритме Метрополиса. В первом (консервативном) случае две случайно выбранные соседние клетки *обмениваются* своими состояниями, в результате число частиц в каждом из двух состояний остается неизменным. Во втором варианте случайно выбранная частица меняет свое состояние на противоположное. В этом варианте модели Изинга состоянию с минимумом энергии соответствует однородная конфигурация, в которой все частицы имеют одинаковое состояние (рис. 3.11).

9 Рассмотрите антиферромагнитный вариант модели Изинга, в котором энергия взаимодействия двух частиц определяется произведением их состояний, взятым со знаком плюс: $E(x_i, x_j) = +x_i x_j$. В этом случае состоянию с минимальной энергией соответствует «шахматная» конфигурация (рис. 3.12), в которой каждая клетка окружена соседями с противоположным состоянием.

10 Реализуйте в NetLogo метод имитации отжига для решения задачи о n ферзях[7]. В этой задаче имеется шахматная доска размера $n \times n$. Требуется расставить n ферзей на этой доске так, чтобы они не били друг друга. Два ферзя бьют друг друга, если они стоят на одной горизонтали, вертикали или диагонали. Чтобы применить к этой задаче метод имитации отжига, необходимо определить понятие энергии взаимодействия ферзей (частиц). Энергию взаимодействия двух ферзей положим равной 1, если они бьют друг друга, равной 0 в противном случае. Полная энергия системы складывается из энергий всех попарных взаимодействий. Таким образом, состоянию с минимумом энергии соответствует решение задачи (ни одна из фигур не бьет другую).

[7] J. Bell, B. Stevens, *A survey of known results and research areas for n-queens*, Discrete Mathematics, Vol. 309, Issue 1, 2009, p. 1–31.

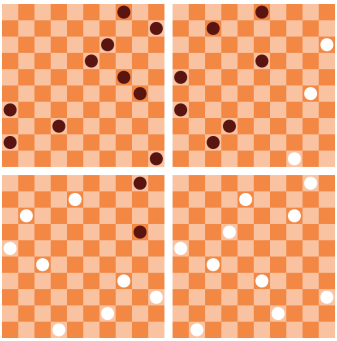


РИС. 3.13 Применение метода имитации отжига к решению задачи о n ферзях

В начальный момент времени k-й ферзь располагается в случайной клетке в k-м горизонтальном ряду доски. Выбор пробного состояния заключается в случайном переносе по горизонтали одной случайно выбранной фигуры. На рис. 3.13 приведен пример процесса решения этой задачи для случая $n = 10$ (темным цветом выделены фигуры под боем).

11 Примените метод имитации отжига к решению задачи коммивояжера[8]. В этой задаче имеется n городов, коммивояжер должен посетить хотя бы по одному разу каждый город и вернуться в первый город. Требуется найти маршрут минимальной длины, удовлетворяющий перечисленным условиям. Эта задача относится к классу NP-сложных задач, т.е. ее точное решение в общем случае требует перебора экспоненциального числа решений[9]. Реализуйте эту задачу в NetLogo. Города в модели представляются черепахами, положение которых выбирается случайным образом. Каждой черепахе приписывается уникальный идентификатор `id`. Текущий маршрут представляется некоторой перестановкой всех идентификаторов, которую можно реализовать, используя атрибуты черепах `prev` и `next` — идентификаторы предшествующей и последующей черепах в заданном маршруте. Целевой функцией в этой задаче является длина маршрута. Для выбора пробного решения можно использовать несколько схем (рис. 3.14). Для простой схемы случайным образом выбираются два соседних в текущем маршруте города, после чего они меняются местами в этом маршруте. В остальных вариантах выбираются два случайных города, для которых и выполняется показанное на рис. 3.14 действие. Для рисования текущего маршрута можно использовать вспомогательную черепаху, которая передвигается от города к городу, используя их атрибут `next`. На рис. 3.15 приведен пример работы метода имитации отжига для задачи с $n = 30$ городами, генерация пробных решений проводилась методом инверсии.

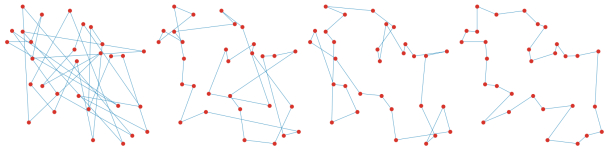


РИС. 3.15 Пример решения задачи коммивояжера для случая $n = 30$ методом имитации отжига

[8] G. B. Dantzig, R. Fulkerson, S. M. Johnson, *Solution of a large-scale traveling salesman problem*, Operations Research, 1954, 2 (4), p. 393–410.

[9] Полное число вариантов в задаче коммивояжера для n городов равно $(n-1)!$ — это число перестановок всех городов, кроме первого.

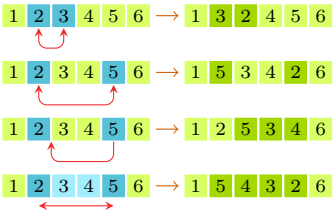


РИС. 3.14 Схемы генерации пробных перестановок (сверху вниз: простая, обмен, вставка, инверсия)