ЗАДАНИЕ 9
Пчелиный алгоритм

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

B Добавляем к интерфейсу модели слайдер с именем targets-number, который будет отвечать за количество целей в модели. Минимальное значение устанавливаем равным 1, максимальное — 20, значение по умолчанию — 10.

C Создаем новый вид черепах — цели (targets во множественном числе, target в единственном числе).

D Добавляем кнопку setup. Пишем код процедуры `setup`, в которой:

- очищаем мир;
- создаем заданное количество целей, для каждой из которых вызываем процедуру настройки `setup-target`;
- вызываем процедуру `color-patches` для раскрашивания патчей;
- сбрасываем таймер.

E Пишем код процедуры `setup-target`, в котором устанавливаем размер цели равным 6, форму — крестик (`"x"`), цвет — белый. Помещаем цель в случайную точку модели.

F Создаем функцию `eval [x y]`, которая будет вычислять целевую функцию для точки с заданными координатами. Функция вычисляет расстояние от этой точки до ближайшей цели:

```
min [distancexy x y] of targets.
```

G Приписываем патчам атрибут `pval` — значение целевой функции в соответствующей точке. Создаем процедуру `color-patches`, в которой сначала просим все патчи вычислить значение атрибута `pval`. Вторым шагом находим максимальное значение `max-val` этого атрибута среди всех патчей. Наконец, просим патчи установить цвет с помощью вызова функции

`my-scale-color (pval / max-val)`,

аргументом которой является нормированное значение атрибута `pval`.

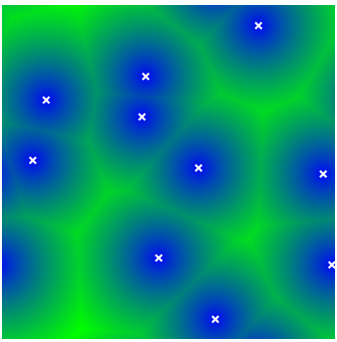

РИС. 9.1 Вид модели с установленными целями

H Создаем функцию `my-scale-color`, в которой аргумент `x` из диапазона от 0 до 1 преобразуется в цвет в формате RGB: от синего (`x = 0`) до зеленого (`x = 1`):

```
to-report my-scale-color [x]
  set x x ^ 0.75
  report rgb 0 (255 * x) (255 - 255 * x)
end
```

I Проверяем работу кнопки setup (рис. 9.1).

J Заставим цели перемещаться. Для этого к интерфейсу модели добавим слайдер target-vel, обозначающий скорость движения целей. Устанавливаем диапазон от 0 (цели неподвижны) до 20, значение по умолчанию — 1, шаг — 0.1. Создадим кнопку go и соответствующую процедуру, в которой сначала попросим все цели передвинуться вперед на расстояние `targets-vel`, затем перекрасим патчи командой `color-patches` и в конце обновим таймер. Проверяем работу кнопки go (рис. 9.2).

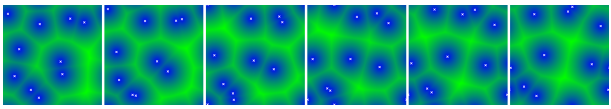

РИС. 9.2 Движение целей

K Создаем новый вид черепах — места (places, place), для представления найденных пчелами мест для более детального поиска. Добавляем к интерфейсу два слайдера places-number (диапазон от 1 до 100) и place-size (диапазон от 1 до 40), соответствующие параметрам k и δ пчелиного алгоритма.

L В процедуре `setup` создаем заданное количество мест, для каждого из которых вызываем процедуру настройки `setup-place`. Создаем процедуру `setup-place`, в коде которой устанавливаем размер равным 6, форму — прямоугольник (`"rect"`), цвет — белый. Помещаем место в случайную точку модели.

M В стандартных формах черепах в NetLogo нет нужной нам формы (незакрашенный квадрат, как на рис. 9.4). Можно создать свою форму (рис. 9.3), используя диалог Tools → Turtle Shapes Editor → New. Выбираем незакрашенный прямоугольник, рисуем квадрат максимального размера (область рисования соответствует одному патчу модели), снимаем переключатель Rotatable, устанавливаем имя rect, нажимаем Ok.

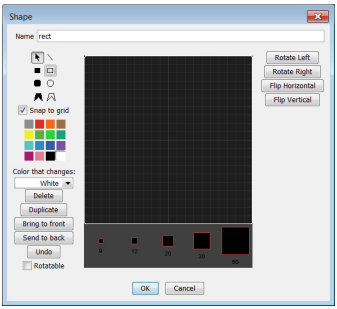

РИС. 9.3 Диалог для создания новых форм черепах

N Проверяем еще раз работу кнопки setup, теперь в модели должны отображаться выбранные случайным образом места (рис. 9.4).

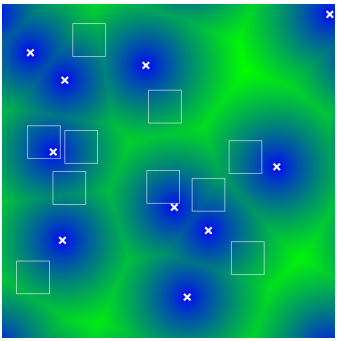

РИС. 9.4 Вид модели со случайно выбранными местами

O К интерфейсу добавим два слайдера: число рабочих пчел, выполняющих поиск в окрестности выделенных мест (foragers-number), и число разведчиков, ведущих свободный поиск по всему пространству модели (scouts-number). Создадим новый вид черепах — пчел (bees, bee). Припишем этому виду два атрибута — `val`, значение целевой функции в той точке, где находится данная пчела; и `my-place`, место, в окрестности которого выполняется поиск. Второй атрибут нужен для обновления координат каждого места по результатам поиска в его окрестности, для пчел-разведчиков этот атрибут не будет использоваться.

P Переходим к написанию самого пчелиного алгоритма (в упрощенной форме, без деления мест на простые и элитные), код которого должен быть помещен в процедуре `go` непосредственно перед командой `tick`. Первым шагом просим все места (places) обновить значение атрибута `val`, вычислив значение целевой функции в точке со своими координатами `xcor` и `ycor`. Удаляем из модели всех пчел, которых мы разместили на предыдущем шаге: `ask bees [die]`.

Q Если число рабочих пчел больше нуля, то про-

сим каждое место создать заданное число пчел (с помощью команды `hatch-bees`, каждая пчела при этом помещается в ту же точку, где располагается данное место) и для каждой пчелы выполнить следующие действия:

- установить атрибут `my-place` равным `myself`;
- к каждой координате пчелы прибавить случайное число из диапазона от `-0.5 * p-size` до `0.5 * p-size`;
- вычислить значение атрибута `val` для своего нового положения;
- установить размер 1.5, форму — круг, цвет — желтый.

После создания и размещения всех пчел для данного места нужно найти пчелу с минимумом целевой функции, и если это минимальное значение окажется меньше, чем значение атрибута `val` для текущего места, то переместить место в точку, где находится эта лучшая пчела, используя команду `move-to`:

```
1 let mates bees with [my-place = myself]
2 let best min-one-of mates [val]
3 if [val] of best < val [move-to best]
```

В первой строке мы создаем набор пчел, привязанных к данному месту. Во второй строке находим лучшую пчелу с минимумом атрибута `val`. В третьей строке обновляем положение текущего места. Проверяем работу модели (кнопка `go`), пример движения места по направлению к ближайшей цели показан на рис. 9.5.

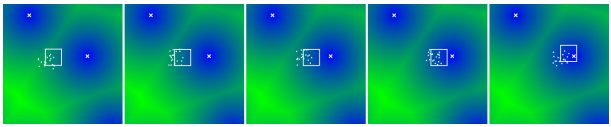

РИС. 9.5 Процесс движения мест в модели

R Если проанализировать поведение модели с одной целью и одним местом, то при увеличении скорости движения цели обновление координат место перестанет успевать реагировать на перемещение цели (проверьте!). Решением этой проблемы является использование пчел-разведчиков, которые выполняют случайный поиск по всей области моде-

ли, а не в локальной окрестности некоторого места. Для этого в процедуре `go`, после описанного выше локального поиска, создаем заданное количество пчел-разведчиков, каждую из которых просим выполнить следующие действия:

- переместиться в случайную точку модели;
- вычислить значение атрибута `val` для своего нового положения;
- установить размер 2.5, форму — круг, цвет — желтый;
- найти место `worst` с самым большим значением атрибута `val`, т.е. худшее из всех мест. Если значение целевой функции у данной пчелы-разведчика меньше (т.е. лучше) аналогичного значения у места `worst`, то попросить `worst` переместиться в точку, где располагается данный разведчик, и перевычислить атрибут `val`.

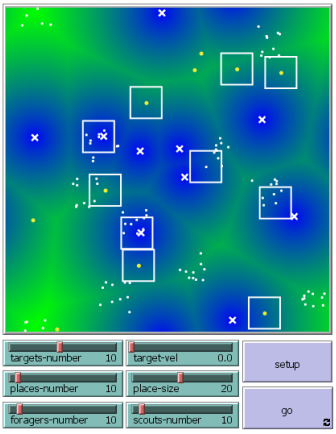

РИС. 9.6 Окончательный интерфейс модели

S Окончательный интерфейс модели показан на рис. 9.6. Проверяем ее работу с неподвижными целями, нулевым числом рабочих пчел и ненулевым числом пчел-разведчиков (рис. 9.7).

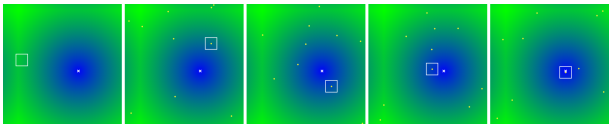

РИС. 9.7 Процесс поиска цели пчелами-разведчиками

УПРАЖНЕНИЯ

1 Будем называть расстоянием от множества мест до цели расстояние до этой цели от ближайшего к ней места. Добавьте к интерфейсу модели график, показывающий зависимость от времени расстояния до самой удаленной от множества мест цели. На рис. 9.8 приведен пример такого рода зависимости для пяти целей и десяти мест. Резкие изменения на графике соответствуют захвату или потере цели колонией пчел.

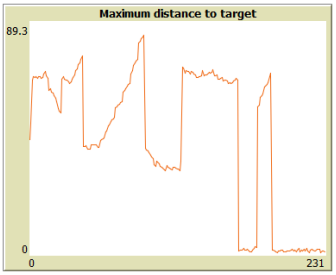

РИС. 9.8 Зависимость от времени расстояния до самой удаленной от множества мест цели

2 Для включения и выключения просмотра отдельных элементов модели (целей, мест, пчел) добавьте к интерфейсу модели соответствующие переключатели.

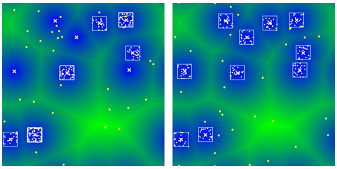

РИС. 9.9 Результат работы модели без учета перекрытия мест (слева, часть целей не покрыта местами) и с учетом этого перекрытия (справа, все цели обнаружены)

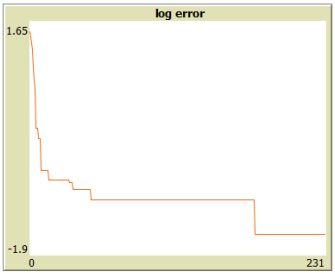

РИС. 9.10 Сходимость метода с фиксированным размером мест (верхний график) и с адаптивным изменением размеров (нижний график), по вертикали используется логарифмический масштаб

3 Примените пчелиный алгоритм к решению задачи одномерной минимизации, взяв за основу модель для метода имитации отжига.

4 Примените пчелиный алгоритм к задаче двумерной оптимизации, взяв за основу модель для метода роя частиц.

5 Включите в модель пчелиного поиска поддержку двух видов мест — элитных и простых.

6 Обычной ситуацией в работе пчелиного алгоритма является наложение (или перекрытие) нескольких мест друг на друга, что является причиной потери разнообразия в колонии пчел. Исключить такого рода ситуации можно, например, следующим образом. Каждое место после обновления своих координат вычисляет расстояние до ближайшего к нему другого места. Если это расстояние меньше половины размера места (т.е. имеется перекрытие), то худшее из этих двух мест заменяется на случайно выбранное место. Проведите эксперимент с неподвижными целями, в котором число мест совпадает с числом целей (рис. 9.9).

7 Рассмотренный в главе вариант алгоритма с фиксированным размером мест обладает следующим существенным недостатком, проявляющимся при решении задач непрерывной оптимизации, в которых важна точность результата. После того как точка глобального экстремума оказывается покрытой каким-нибудь местом, дальнейшая работа алгоритма сводится, по сути, к простейшему случайному поиску в области фиксированного размера, т.к. само место фактически перестает передвигаться. Это очень негативно сказывается на сходимости метода (верхний график на рис. 9.10). Возможным решением этой проблемы является постепенное уменьшение размера мест `place-size`, своего рода имитация отжига по данному параметру. Другим, адаптивным вариантом является уменьшение размера места только в том случае, если оно перекрывается с каким-нибудь другим местом. На нижнем графике на рис. 9.10 показана сходимость такой вариации метода, когда уменьшение размера сводится к умножению на фиксированную константу (0.9 в данном случае).

8 Чтобы более наглядно продемонстрировать адаптивные свойства пчелиного алгоритма, рассмотрите целевую функцию, являющуюся суммой двух перевернутых гауссианов:

$$f(x,t) = -\sum_{k=1}^{2} \alpha_k(t) e^{-\eta_k \cdot \|x - x_k\|}, \qquad (9.1)$$

где x_k — центр k-го гауссиана, η_k — его ширина, коэффициенты $\alpha_k(t)$ меняются со временем по следующему закону:

$$\alpha_k(t) = \frac{1 - \sin(t + \pi k).}{2} \qquad (9.2)$$

Таким образом, в каждый момент времени глобальный минимум находится практически в центре одного из гауссианов, ровно половину времени в первом, половину — во втором (рис. 9.11).

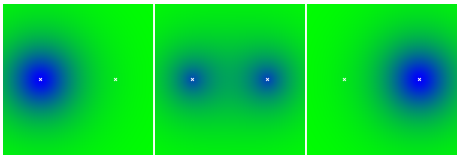

РИС. 9.11 Вид функции (9.1) в разные моменты времени

Исследуйте, как ведет себя модель при различных значениях ее параметров. Например, на рис. 9.12 показано, как происходит переключение колонии бактерий с одного минимума на другой. Верхний график соответствует колонии без пчел-разведчиков, нижний — колонии с разведчиками. Видно, что использование разведчиков существенно сокращает время переключения.

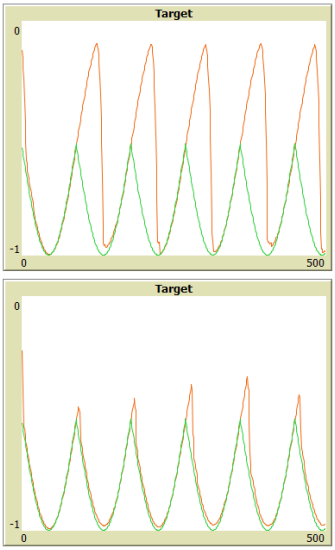

РИС. 9.12 Зависимость целевой функции (9.1) от времени (зеленая линия) и текущее лучшее значение, найденное колонией пчел (красная линия)

9 Разработайте вариант модели из упражнения 8, в котором целевая функция представляет собой некоторый возобновляемый ресурс. Аналогично формуле (9.1) целевая функция является взвешенной суммой произвольного числа перевернутых гауссианов, весовые коэффициенты $\alpha_k(t)$ которых зависят от времени следующим образом:

$$\alpha_k(t+1) = \gamma\alpha_k(t) - \delta m_k(t), \qquad (9.3)$$

где первое слагаемое представляет собой естественный рост ресурса ($\gamma > 1$), второе слагаемое соответствует потреблению этого ресурса пчелами: $m_k(t)$ — количество мест, покрывающих центр соответствующего гауссиана (источника ресурса), δ — скорость потребления ресурса. Исследуйте поведение модели при различных значениях ее параметров, в частности при отсутствии пчел-разведчиков. На рис. 9.13 приведен пример поведения колонии пчел в данной модели, показаны траектории движения мест в процессе поиска новых, более сильных источников ресурса.

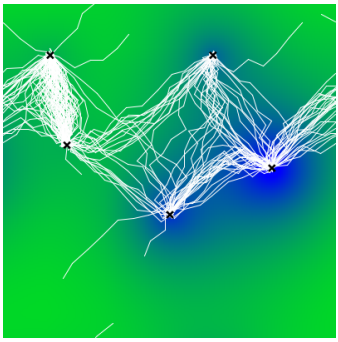

РИС. 9.13 Траектории движения мест в модели с возобновляемым ресурсом

10 Пчелы используют свой танец также при поиске нового места для построения нового гнезда. В этом случае, в отличие от поиска нектара, все пчелы-разведчики

должны прийти к консенсусу, т.е. выбору ровно *одного* места. Для этого пчелами во время танца применяются специальные стоп-сигналы, которые тормозят танцевальную активность других разведчиков[1]. Такой механизм взаимного торможения гарантирует достаточно быстрое принятие единого решения всеми пчелами. В библиотеке моделей NetLogo имеется готовая модель BeeSmart Hive Finding, в которой реализуется данный механизм поведения пчел (рис. 9.14). Проведите численное исследование этой модели.

11 Помимо специфического способа коммуникации в форме пчелиного танца, пчелы демонстрируют и другие типы коллективного поведения. Одним из примеров такого поведения, которое до сих пор не получило своего объяснения, является процесс построения пчелиных сот в форме абсолютно правильной гексагональной (шестиугольной) решетки[2]. Согласно одной из гипотез, пчелы строят на самом деле соты круглой формы, которые после остывания воска принимают форму уже шестиугольников. Простая модель[3] Honeycomb, описывающая процесс возникновения таких структур, также имеется в библиотеке моделей NetLogo. Основная идея модели заключается в том, что параметры движения пчел (угол поворота и скорость) определяются количеством воска в том месте, где находится пчела. Все параметры модели встроены прямо в ее код, для того чтобы исследовать поведение модели, добавьте к ее интерфейсу слайдеры и элементы ввода, связанные с соответствующими параметрами. На рис. 9.15 приведен пример возникновения гексагональной структуры в процессе работы этой модели.

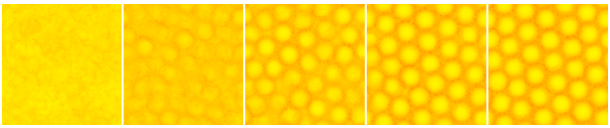

РИС. 9.15 Формирование гексагональной решетки

[1] T. Seeley, P. Visscher et al., *Stop Signals Provide Cross Inhibition in Collective Decision-Making by Honeybee Swarms*, Science, 2012, Vol. 335. p. 108–111.

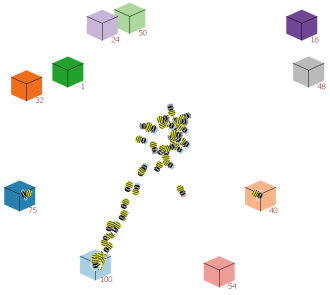

РИС. 9.14 Модель выбора колонией пчел места для нового гнезда

[2] S. Camazine, J. Sneyd et al., *A mathematical model of self-organized pattern formation on the combs of honeybee colonies*, Journal of Theoretical Biology, Vol. 147, 4, 1990, p. 553–571.

[3] Которая, впрочем, пока не имеет никакого экспериментального подтверждения!