Лекция 3. Системы Линденмайера

Системы Линдеммайера, или L-системы, являются частным случаем строковых перезаписывающих систем, некоторым специальным вариантом формальных грамматик. От последних L-системы отличаются параллельным характером применения правил. L-системы были разработаны венгерским биологом Аристидом Линденмайером в 1968 году. Основной областью применения L-систем является моделирование процессов роста и формообразования в различных биологических системах, прежде всего у растений.

0.1 L-системы

В основе построения L-систем лежит идея перезаписи (rewriting), под которой понимается последовательная замена одних частей заданного простого объекта на другие согласно некоторым правилам (называемыми иногда продукциями). Наиболее развитой в настоящее время является теория построения и анализа *строковых перезаписывающих систем*, в которых объектами являются цепочки смиволов в некотором алфавите. Классические примеры такого рода систем — формальные грамматики, разработанные американским лингвистом Хомским в 1950-х годах, и нормальные алгоритмы Маркова, предложенные советским математиком А. А. Марковым примерно в то же самое время. Системы Линденмайера относятся к такому же классу систем, их особенностью является порядок применения правил вывода. Если в формальных грамматиках и алгоритмах Маркова правила применяются строго последовательно, то в L-системах на каждом шаге применяется максимально возможное число правил (принцип максимального параллелизма). Это различие обусловлено «биологическим» происхождением L-систем — отдельные символы в L-системах трактуются как клетки, эволюция которых происходит параллельно.

0.1.1 Определение L-системы

Системой Линденмайера (*L-системой*) называется тройка вида

$$L = (V, \omega_0, R),$$

где V — конечный алфавит системы, $\omega_0 \in V^*$ — непустая цепочка символов над алфавитом V, называемая *аксиомой*, R — набор правил. В стандартном

варианте L-систем правила вывода имеют вид

$$v \to \alpha,$$

где v — некоторый символ заданного алфавита V, $\alpha \in V^*$ — цепочка символов (возможно пустая) в том же алфавите. Таким образом, каждое правило может трактоваться, либо как деление клетки ($|\alpha| > 1$), либо ее модификация ($|\alpha| = 1$), либо как ее смерть ($|\alpha| = 0$).

Если для любого символа заданного алфавита имеется не более одного правила вывода, то такая L-система называется *детерминированной*. Если для каких-то символов имеется более одного правила вывода, то система называется *недетерминированной*. Если применимость всех правил к символам не зависит от окружения этих символов в текущей цепочке (такое окружение называется *контекстом*), то L-система называется *контекстно-свободной*, в противном случае система называется *контекстно-зависимой*.

Наиболее простыми являются детерминированные контекстно-свободные L-системы, называемые кратко D0L-системами.

0.1.2 D0L-системы

Все правила D0L-систем имеют вид $v \to \alpha$, т. е. представляют собой отображение множества символов алфавита системы в множество символьных цепочек над этим алфавитом. Для каждого символа должно быть определено ровно одно правило. В том случае, когда некоторые символы в силу специфики задачи не должны преобразовываться, для них определяются тривиальные правила вида $v \to v$.

Процесс эволюции L-системы выглядит следующим образом. Начальная цепочка представлена аксимой ω_0. На каждом шаге все символы текущей цепочки одновременно заменяются на некоторые подцепочки согласно системе правил. Предполагается, что такая замена производится параллельно, поэтому отдельные шаги эволюции системы ассоциируются с (дискретным) временем.

Рассмотрим простой пример L-системы, алфавит которой содержит два символа a и b, а система правил имеет вид

$$\begin{cases} R_1 : a \to ab \\ R_2 : b \to a. \end{cases}$$

Пусть аксиома состоит из единственного символа $\omega_0 = b$. Тогда, первые несколько шагов эволюции данной системы будут выглядеть следующим образом.

$$b \to a \to ab \to aba \to abaab \to abaababa \to abaababaabaab \to \dots.$$

Если посчитать длины цепочек на каждом шаге эволюции, то получим следующую последовательность $(1, 1, 2, 3, 5, 8, \dots)$, которая является классической последовательностью Фибоначчи: $\varphi_0 = \varphi_1 = 1$, $\varphi_{i+1} = \varphi_i + \varphi_{i-1}$, $i > 1$. Отсюда можно заключить, что длина цепочки данной L-системы растет экспоненциально со временем. Кстати, последовательность Фибоначчи получается и при подсчете количества отдельных символов a или b в цепочке.

Приведем пример D0L-системы, обладающей полиномиальным ростом длины цепочки: алфавит $V = \{a, b\}$, система правил содержит единственное

правило $R_1 : a \to ba$. Процесс применения этого правила к аксиоме $\omega_0 = a$ выглядит следующим образом:

$$a \to ba \to bba \to bbba \to bbbba \to \dots.$$

Очевидно, что длина цепочки в данном случае растет линейно со временем.

Простым примером L-системы, обладающей периодическим поведением, является система с алфавитом $V = \{a, b, c\}$ и набором правил (λ обозначает пустую цепочку)

$$\begin{cases} R_1 : a \to bc \\ R_2 : b \to \lambda \\ R_3 : c \to a. \end{cases}$$

Эволюция такой системы, начиная с аксимомы $\omega_0 = a$, выглядит следующим образом

$$a \to bc \to a \to bc \to a \to \dots.$$

Обобщенный алгоритм моделирования L-системы, заданной системой правил R, может быть описан следующим образом.

LSystem(R, ω_0, n)

```
1: w ← ω_0
2: for t ∈ [1 .. n] do
3:     u ← λ    ▷ пустая строка
4:     for i ∈ [1 .. |w|] do
5:         u ← u + R[w[i]]
6:     w ← u
7:     print w
```

0.1.3 Геометрическая интерпретация

Пусть задан алфавит $T = \{F, f, +, -\}$. Рассмотрим геометрическую интерпретацию цепочек символов над этим алфавитом, основанную на системе команд специального рисующего устройства[1]. В каждый момент времени устройство находится в некоторой точке плоскости, и ориентировано в некотором направлении (начальное направление определяется соглашением, например, в приводимых ниже примерах устройство в начальный момент времени смотрит вправо). Положение устройства и его ориентация образуют *состояние* этого устройства. Символы алфавита T трактуются, как команды этому устройству.

- команда «F» — перемещение устройства на расстояние d в направлении, определяемым текущим состоянием устройства, с прорисовкой линии движения;
- команда «f» — аналогичная команда, но движение производится без прорисовки;
- команда «+» — поворот устройства против часовой стрелки на заданный угол α;

[1] В терминах языка Лого такое устрйоство называется черепашкой

- команда «$-$» — поворот устройства по часовой стрелки на заданный угол α.

Например, выполнение последовательности команд, заданных цепочкой $F-F-F-F-F$ для угла $\alpha = 144^\circ$, приведет к построению правильной пятиконечной звезды (рисунок 0.1).

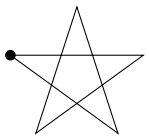


Рис. 0.1 Геометрическая интерпретация цепочки $F-F-F-F-F$

Использование такой графической интерпретации позволяет легко описывать и строить разнообразные самоподобные структуры (фракталы). Например, используя только две команды «F» и «f», можно построить первые несколько приближений к известной фрактальной структуре — канторову множеству. Это делается с помощью простой системы правил

$$\begin{cases} R_1 : F \to FfF \\ R_2 : f \to fff, \end{cases}$$

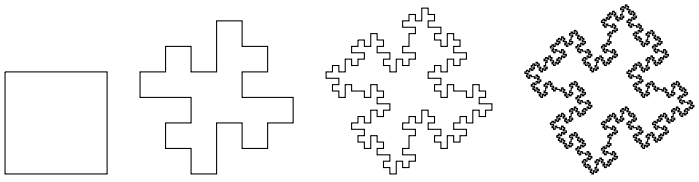
примененной к аксимоме $\omega_0 = F$. Результат (первые шесть итераций) показан на рисунке 0.2.

Рис. 0.2 Канторова пыль

Включение в набор правил символов поворота позволяет перейти к построению двумерных фракталов. Классический пример такого построения — квадратный остров Коха, описываемый единственным правилом

$$R : F \to F-F+F+FF-F-F+F.$$

На рисунке 0.3 приведены первые четыре шага эволюции такой системы для аксиомы $\omega_0 = F-F-F-F$. Длина отрезка на каждом шаге уменьшается в четыре раза по сравнению с предыдущим шагом, так чтобы начальная и конечная точки каждой команды «F» совпадали с начальной и конечной точками получаемых в результате замены ломаных линий $F-F+F+FF-F-F+F$.



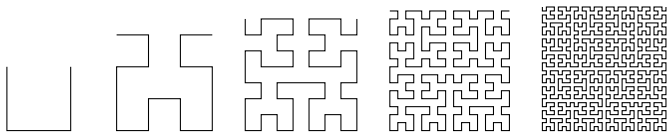
а) $t = 0$ б) $t = 1$ в) $t = 2$ г) $t = 3$

Рис. 0.3 Аксиома и три первых шага при построении квадратного острова Коха

Еще одним примером использования L-систем для генерации самоподобных структур является построение кривых Пеано, которые в пределе заполняют всю заданную плоскую фигуру, чаще всего, квадрат. На рисунке 0.4 показан пример построения кривой Гильберта, которое выполняется с помощью системы правил

$$\begin{cases} A \to +BX - AXA - XB+ \\ B \to -AX + BXB + XA-, \end{cases}$$

примененной к аксиоме $\omega_0 = A$. Заметим, что в этой L-системе используется интерпретация символов слегка отличная от той, которая использовалась нами выше. Символы A и B здесь означают перемещение на заданное расстояние с прорисовкой линии (т. е. аналогичны команде F), а символу X не соответствует никакая графическая команда, этот символ используется только для управления ходом роста системы и игнорируется рисующим устройством.



а) $t = 1$ б) $t = 2$ в) $t = 3$ г) $t = 4$ д) $t = 5$

Рис. 0.4 Пять первых шагов при построении кривой Гильберта

Описанная геометрическая интерпретация может быть расширена и на случай трех измерений. Для этого вводится новое понятие текущей ориентации устройства, которое описывается тремя векторами H, L и U, задающими направление «вперед», «влево» и «вверх». По сути, это ортонормированная система координат, привязанная к рисующему устройству. Повороты устройства в пространстве описываются формулой

$$[H', L', U'] = [H, L, U] \cdot R,$$

где R — одна из трех матриц поворота на угол α вокруг каждого из трех базисных веторов:

$$R_H(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_L(\alpha) = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

$$R_U(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

Для каждого базисного вектора вводятся две команды поворота на угол δ:

- «+» — поворот влево, используя матрицу $R_U(\delta)$;

- «$-$» — поворот вправо, используя матрицу $R_U(-\delta)$;
- «d» — наклон вперед на угол δ, используя матрицу $R_L(\delta)$;
- «u» — наклон назад на угол δ, используя матрицу $R_L(-\delta)$;
- «l» — наклон влево на угол δ, используя матрицу $R_H(\delta)$;
- «r» — наклон вправо на угол δ, используя матрицу $R_H(-\delta)$.

0.1.4 Моделирование роста растений

Используя предложенные наборы графических команд, можно строить весьма разнообразные объекты, однако, любой такой объект всегда будет оставаться одномерным, т. е. представлять собой некоторую ломаную кривую на плоскости или в пространстве, у которой часть звеньев прорисована (команда F), а часть нет (команда f). Вместе с тем, основным предназначением L-систем при их создании было моделирование процессов роста у растений, а для всех растений характерной чертой роста является *ветвление*. С точки зрения теории графов, ветвление приводит к образованию древовидных структур данных, более конкретно, корневых деревьев. Корневое дерево, это связанный неориентированный ациклический граф, у которого выделена одна из вершин, называемая корнем дерева.

Оказалось, что моделирование процесса ветвления возможно с помощью обычных строковых L-систем. Основная идея заключается в том, что существует взаимно-однозначное соответствие между корневыми деревьями и правильными скобочными записями. Внешняя пара скобок в такой записи соответствует корню дереву, все пары скобок, непосредственно вложенные во внешнюю пару скобок, образуют дочерние вершины для корня и т. д. Скобки, не содержащие внутри себя других скобок, образуют листья дерева. На рисунке 0.5 показано дерево, соответствующее скобочной записи $[_1[_2]_2[_3[_4]_4[_5]_5]_3]_1$ (номера скобкам расставлены для удобства восприятия данной записи).

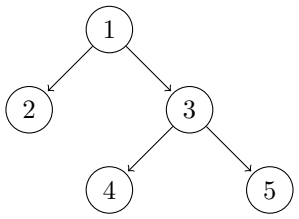


Рис. 0.5 Дерево, соответствующее записи $[_1[_2]_2[_3[_4]_4[_5]_5]_3]_1$

Для организации ветвления в язык рисующего устройства вводятся две новые команды, обозначаемые символами открывающей и закрывающей скобок «[» и «]». Кроме того, само устройство оснащается стеком, куда оно может записывать свои состояния. Две новые команды являются стандартными стековыми командами *push* и *pop*:

- «[» — записать в стек текущее состояние;
- «]» — извлечь из стека состояние и сделать его текущим.

Напомним, что под состоянием устройства понимается его местоположение (например, координаты) и ориентация (направление движения, задаваемое вектором). Таким образом, каждый символ «[», встречаемый в строке, означает по сути пометку — вернуться в данное место после того, как будет встречен соответствующий символ «]». На рисунке 0.5 приведен пример интерпретации последовательности команд $F[+F][-F[-F]F]F[+F][-F]$.

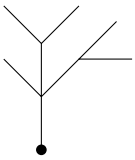


Рис. 0.6 Геометрическая интерпретация цепочки $F[+F][-F[-F]F]F[+F][-F]$

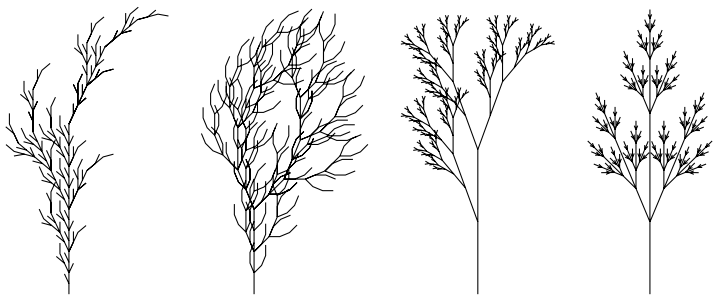
Варьируя правила L-системы и параметры рисующего устройства (длина сегмента и угол поворота), можно получать весьма разнообразные «растительные» структуры. Опять же, в алфавит L-системы можно включать дополнительные управляющие процессом роста символы, которые игнорируются устройством. На рисунке 0.7 приведено несколько примеров построения древовидных структур с помощью следующих L-систем (заданы система правил, аксиома ω_0, угол δ и число поколений n).

а) $\begin{cases} R_1 : F \to F[+F]F[-F] \\ \omega_0 = F, \delta = 25°, n = 4; \end{cases}$

б) $\begin{cases} R_2 : F \to FF - [-F + F + F] + [+F - F - F] \\ \omega_0 = F, \delta = 22.5°, n = 3; \end{cases}$

в) $\begin{cases} R_1 : X \to F[+X]F[-X] + X \\ R_2 : F \to FF \\ \omega_0 = X, \delta = 20°, n = 6; \end{cases}$

г) $\begin{cases} R_1 : X \to F[+X][-X]FX \\ R_2 : F \to FF \\ \omega_0 = F, \delta = 22.5°, n = 6. \end{cases}$



а) б) в) г)

Рис. 0.7 Примеры построения древовидных структур

0.2 Вариации L-систем

D0L-системы являются простым и удобным средством моделирования, но выразительные возможности этих систем являются все-таки относительно бедными. Поэтому, были разработаны разнообразные вариации L-систем, из которых наиболее важными являются стохастические (недетерминированные), контекстно-зависимые и параметрические L-системы.

0.2.1 Стохастические системы

Очевидным недостатком D0L-систем с точки зрения моделирования процессов формообразования (морфогенеза) является то, что система правил, аксиома и параметры рисующего устройства *однозначно* определяют внешний вид генерируемой структуры. В реальности ситуация является отличной от этого — все растения внутри одного вида хотя и обладают схожей структурой, но могут сильно отличаться в деталях. Это различие определяется прежде всего тем, что на рост растений не в последнюю очередь влияет среда. Т. к. такое влияние, как правило, носит стохастический характер, то внесение некоторой доли недетерминизма в L-системы должно положительно сказаться на их моделирующих свойствах.

Стохастической L-системой называется четверка вида

$$L = (V, \omega_0, R, P),$$

где V — конечный алфавит системы, $\omega_0 \in V^*$ — аксиома, R — набор правил, $P : R \to (0, 1]$ — функция вероятности, ставящая каждому правилу из R вероятность его применения. На функцию P накладывается ограничение, что сумма вероятностей всех правил с одинаковой левой частью должна быть равна 1 (или не превосходит 1, т. к. при необходимости, всегда можно дополнить систему правил тривиальным правилом вида $v \to v$). Для правила с вероятностью применения p используется следующее обозначение: $v \xrightarrow{p} \alpha$.

Простым примером стохастической L-системы может служить следующая система:

$$\begin{cases} R_1 : F \xrightarrow{0.4} F[+F]F[-F]F \\ R_2 : F \xrightarrow{0.3} F[+F]F \\ R_3 : F \xrightarrow{0.3} F[-F]F. \end{cases}$$

На рисунке 0.8 показано несколько случайных структур, сгенерированных с помощью этой системы из одной и той же аксиомы $\omega_0 = F$.

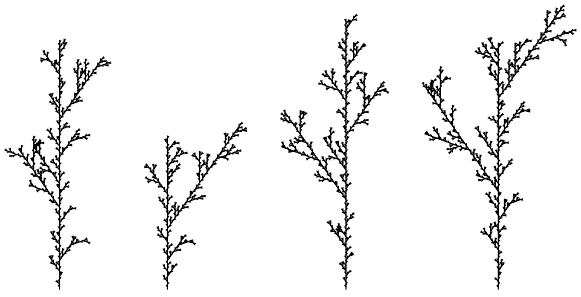


Рис. 0.8 Примеры построения случайных древовидных структур

0.2.2 Контекстно-зависимые системы

В реальных биологических системах каждая клетка организма несет один и тот же набор генов, управляющих ее поведением. Специализация клеток под различные типы поведения часто определеляется непосредственным окружением этой клетки, т. е. ее *контекстом*. В D0L-системах контекст символов никак не учитывается, т. е. каждый символ развивается по сути независимо от других символов. Например, если вырезать из текущей строки некоторую подстроку, то эта вырезанная подстрока будет развиваться точно также, как она развивалась бы внутри исходной строки.

Чтобы учесть влияние символов друг на друга, нужно ввести контекст в правила L-системы. Контекст символа может либо правым, либо левым, либо двусторонним. Если все правила обладают только односторонним контекстом (одного типа), то такая L-система называется 1L-системой. Если же некоторые правила обладают двусторонним контекстом, то система называется 2L-системой.

Правила, учитывающие контекст, задаются обычно следующим образом:

$$L\langle v\rangle R \to \alpha,$$

где $L \in V^*$ — левый контекст символа v, $R \in V^*$ — правый контекст этого символа. Применение этого правила приводит к замене символа v на цепочку α, при этом контекст символа не меняется (символы, входящие в контекст меняются по своим правилам).

Заметим, что все рассмотренные выше детерминированные L-системы можно обобщить до так называемых (k, l)-систем: каждое правило в таких системах обладает левым контекстом длины не более k символов, правым — длины не более l символов. Например, D0L-системы в данной классификации будут называться $(0, 0)$-системами.

В качестве примера контекстно-зависимой L-системы можно рассмотреть простую $(1, 0)$-систему, моделирующую распространение сигнала:

$$\begin{cases} b\langle a \to b \\ b \to a. \end{cases}$$

Например, первые пять шагов развития данной системы, примененной к аксиоме $\omega_0 = baaaaaaaa$, выглядят следующим образом:

$$\begin{aligned} t = 0: &\quad baaaaaaaa \\ t = 1: &\quad abaaaaaaa \\ t = 2: &\quad aabaaaaaa \\ t = 3: &\quad aaabaaaaa \\ t = 4: &\quad aaaabaaaa. \end{aligned}$$

0.2.3 Параметрические системы

В стандартных L-системах, рассмотренных выше, любые два одинаковых символа являются неотличимыми друг от друга. Это означает, что их геометрическая интерпретация будут одинаковой. Например, символ F всегда (в рамкой заданной L-системы) будет изображаться отрезком одинаковой длины. Чтобы построить отрезок другой длины, придется составлять его из нескольких стандартных отрезков. Если желаемая длина не делится

нацело на длину стандартного отрезка, то прижется уменьшать стандартную длину. Понятно, что такого рода процедуры сильно усложняют процесс моделирования, большая часть правил в этом случае будет отвечать за различные технические аспекты моделирования. Это является одной из причин введения в рассмотрение параметрических L-систем.

В *параметрических* L-системах каждому символу используемого алфавита приписывается несколько числовых параметров. Каждое правило L-системы определяет помимо прочего и значения параметров символов, возникающих в результате применения этого правила. Кроме того, любому правилу может быть поставлено некоторое условие применимости, зависящее от параметров символов, стоящих в левой части правила.

Простейший пример параметрической D0L-системы состоит из единственного правила: $F(x) \to F(2x)$. Символу F, означающему прямолинейный отрезок, приписан параметр x, являющийся длиной этого отрезка. Применение этого правила к аксиоме $\omega_0 = F(1)$ приводит к экспоненциальному росту длины единственного сегмента:

$$F(1) \to F(2) \to F(4) \to F(8) \to \dots.$$

Такая система с геометрической точки зрения является эквивалентной D0L-системе вида $F \to FF$. Важнейшее отличие заключается в том, что в параметрическом варианте в любой момент времени имеется единственный символ с одним параметром, а в стандартном варианте число символов растет экспоненциально.

Использование условий позволяет контролировать изменение параметров. Например, можно сконструировать L-систему (аналогичную в чем-то только что рассмотренным), в которой рост длины будет ограничен заданной величиной:

$$\begin{cases} F(x) : x < l \to F(x+1) \\ F(x) : x = l \to F(x+1)F(1). \end{cases}$$

Например, для значения $l = 2$ и аксиомы $\omega_0 = F(1)$ первые несколько итераций будут выглядеть следующим образом:

$$F(1) \to F(2) \to F(3)F(1) \to F(3)F(2) \to \\ \to F(3)F(3)F(1) \to F(3)F(3)F(2) \to F(3)F(3)F(3)F(1) \to \dots.$$

0.3 Параллельная реализация L-систем

L-системы любого из рассмотренных типов обладают высокой степенью встроенного параллелизма, т. к. все символы текущей цепочки могут (а в идеале — должны) обрабатываться параллельно. Особенно просто дело обстоит с D0L-системами, для которых выполняется следующее свойство: развитие любой подцепочки заданной цепочки будет точно таким же, если ее рассматривать изолированно.

Эти замечания делает очевидной следующую параллельную реализацию L-системы. Разделим текущую цепочку на некоторое количество подцепочек (примерно одинаковой длины) и поместим каждую полученную подцепочку на отдельный процессорный узел. На каждом шаге эволюции отдельные узлы вычислительной системы в общем случае сначала обмениваются контекстной информацией. Если рассматривается (k, l)-система, то

два узла p_1 и p_2, на которых обрабатываются две смежные подцепочки, должны обменяться информацией по следующей схеме: узел p_1 пересылает свои k крайних левых символов узлу p_2, после чего узел p_2 пересылает свои l крайних правых символов узлу p_1. Эта процедура выполняется для всех пар смежных узлов, после чего каждый узел сможет выполнить один шаг эволюции для своей подцепочки. Заметим, что для D0L-систем стадия обмена контекстной информацией не нужна.

В пределе каждый узел может обрабатывать по одному символу, но такая схема скорее всего будет неоправданной, т. к. вычислительная составляющая работы каждого узла по отношению к обмену информацией будет очень небольшой. Хотя такой низкоуровневый параллелизм может быть эффективным, например, при реализации на GPU.

Предложенная схема параллельной реализации имеет следующий очевидный недостаток. Как правило, аксиома, с которой начинается развитие L-системы, представляет собой относительно короткую цепочку символов, очень часто состоящую вообще из единственного символа. Это означает, что данная цепочка не может быть изначально распределена на несколько процессорных узлов, и скорее всего будет помещена на единственный узел. Т.к. рост L-систем в большинстве случаев является очень быстрым, то уже через несколько шагов эволюции цепочка символов станет достаточно длинной для того, чтобы поместить ее на несколько узлов. Соответственно возникает задача управления динамической загрузкой системы.

Такое управление может быть *централизованным*, когда на каждом шаге вычисляется новая длина цепочки, определяются длины подцепочек (исходя из имеющегося в распоряжении числа процессоров) и производится перераспределение символов. Понятно, что такая схема управления будет очень накладной в смысле коммуникационной нагрузки, кроме того, эта схема предполагает общую синхронизацию всей системы, что также может привести к снижению общей производительности.

Альтернативой может служить какая-нибудь *локальная адаптивная* схема управления нагрузкой. Например, два смежных узла на каждом i-ом шаге выравнивают свою нагрузку, обмениваясь данными. Такой способ не требует общей синхронизации и использования специальных мастер-процессов, управляющих загрузкой системы. Кроме того, процесс выравнивания нагрузки можно совместить с процессом обмена контекстной информацией.

Литература

[1] S. Achasova, O. Bandman, V. Markova, et al., *Parallel Substitution Algorithm. Theory and Application.*, Singapore: World Scientific, 1994.

[2] L. M. Adleman, *Molecular Computation Of Solutions To Combinatorial Problems*, Science, 266, 11, pp. 1021–1024, 1994.

[3] E. Alba, B. Dorronsoro, *Cellular Genetic Algorithms*, Springer, 2008.

[4] D. Boneh, C. Dunworth, R. J. Lipton, J. Sgall, *On the Computational Power of DNA*, DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, 71, 1996.

[5] C. Detrain, J. L. Deneubourg, *Self-Organized Structures in a Superorganism: Do Ants Behave Like Molecules?*, Physics of Life Reviews 3, no. 3, pp. 162-187, 2006.

[6] M. Dorigo, M. Birattari, T. Stutzle, *Ant Colony Optimization*, Technical Report No. TR/IRIDIA/2006-023, September 2006.

[7] S. Goss, S. Aron, J.-L. Deneubourg, J. M. Pasteels, *Self-organized shortcuts in the Argentine ant*, Naturwissenschaften, vol. 76, pp. 579–581, 1989.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[9] J. Kennedy, R. C. Eberhart, *Particle swarm optimization*, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.

[10] K. M. Passino, *Biomimicry of bacterial foraging for distributed optimization and control*, IEEE Control Systems Magazine, 22, pp. 52–67, 2002.

[11] Gh. Paun, *Computing with Membranes*, Journal of Computer and System Sciences, 61, 1 (2000), 108-143.

[12] Gh. Paun, *P Systems with Active Membranes: Attacking NP Complete Problems*, CDMTCS Research Report Series, CDMTCS-102, May 1999.

[13] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, *The Bees Algorithm — A Novel Tool for Complex Optimisation Problems*, Proceedings of IPROMS 2006 Conference, pp. 454–461, 2006.

[14] P. Prusinkiewicz, A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1996 (http://algorithmicbotany.org/papers/abop)

[15] C. W. Reynolds, *Flocks, herds and schools: a distributed behavioral model*, Computer Graphics, 21, 4, pp. 25-34.

[16] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1980

[17] T. Stutzle, H. Hoos, *MAX–MIN Ant System*, Future Generation Computer Systems, vol. 16, no. 8, pp. 889–914, 2000.

[18] *The Oxford handbook of membrane computing*, edited by G. Paun, G. Rozenberg and A. Salomaa, Oxford University Press, 2010.

[19] T. Weise, *Global Optimization Algorithms – Theory and Application*, http://www.it-weise.de/.

[20] D. Whitley, *A Genetic Algorithm Tutorial*, Statistics and Computing (4): 65-85, 1994.

[21] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

[22] Hsu-Chun Yen, *Introduction to Petri Net Theory*, Recent Advances in Formal Languages and Applications, 2006, pp. 343-373.

[23] М. Гарднер, *Крестики–нолики*, М.: Мир, 1988

[24] В. Котов, *Сети Петри*, М.: Наука, 1984.

[25] А. А. Марков, Н. М. Нагорный, *Теория алгорифмов*, М.: Наука, 1984.

[26] М. Минский, С. Пейперт, *Персептроны*, М.: Мир, 1971.

[27] Дж. фон Нейман, *Теория самовоспроизводящихся автоматов*, М.: Мир, 1971

[28] Г. Паун, Г. Розенберг, А. Саломаа, *ДНК-компьютер. Новая парадигма вычислений*, М.: Мир, 2003.

[29] Дж. Питерсон, *Теория сетей Петри и моделирование систем*, М. : Мир, 1984.

[30] Ф. Розенблатт, *Принципы нейродинамики: Перцептроны и теория механизмов мозга*, М.: Мир, 1965.

[31] Т. Тоффоли, Н. Марголус, *Машины клеточных автоматов*, М.: Мир, 1991

[32] Ф. Уоссермен, *Нейрокомпьютерная техника: Теория и практика*, М.: Мир, 1992.

[33] С. Хайкин, *Нейронные сети: Полный курс*, М.: «Вильямс», 2006.