1 ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ И ПРОГРАММИРОВАНИЯ

АЛГОРИТМЫ УКЛАДКИ ДИАГРАММ СОСТОЯНИЙ

М.А. Коротков
Научный руководитель – к.ф.-м.н., доцент Ф.А. Новиков

В статье приведен обзор методов плоской укладки графов, поставлена задача плоской укладки диаграмм состояний *UML*. Описан алгоритм с физической моделью (метод отжига) и один из аналитических алгоритмов.

Введение

Программный пакет с открытым исходным кодом *UniMod* [1] обеспечивает разработку и выполнение автоматно-ориентированных программ. Он позволяет создавать и редактировать диаграммы классов и состояний *UML* [2], которые соответствуют графу переходов и схеме связей конечного автомата [3]. После создания диаграмм существует возможность выполнить их, при этом содержимое диаграмм преобразуется в *XML*-описание, которое передается интерпретатору, также входящему в пакет *UniMod*. Такой подход является реализацией парадигмы автоматного программирования [4].

Во многих современных редакторах для текстовых языков программирования существует возможность автоматического форматирования программного кода. Если программным кодом является не текст, а набор диаграмм, задача форматирования текста преобразуется в задачу укладки диаграмм. Более подробно задача будет описана ниже. В рамках проекта *UniMod* основной интерес представляет укладка диаграмм состояний языка *UML*.

При укладке диаграммы необходимо учитывать ряд критериев, которые могут противоречить друг другу, например, критерии минимизации площади, занимаемой диаграммой, и наличия достаточного количества свободного места («воздуха»). Каждый критерий оценивает «качество» диаграммы для того или иного применения (отображения на мониторе, печати т.д.). Такие критерии называются **эстетиками**. Задавшись некоторым набором эстетик, можно построить штрафную функцию (которая тем больше, чем больше расхождения между изображением диаграммы и критериями) и пытаться минимизировать её, перемещая элементы, или разработать алгоритм, последовательно модифицирующий исходную диаграмму так, чтобы результат соответствовал выбранным критериям.

1. Постановка задачи

Рассмотрим диаграмму состояний конечного автомата. Она включает в себя состояния и переходы (и у тех, и у других есть дополнительные, связанные с ними, сущности – метки, но мы сейчас не будем останавливаться на этом вопросе). Диаграмме состояний (без вложенных состояний) можно сопоставить граф [5] (при укладке диаграммы состояний нам не важна ориентация дуг). Каждому состоянию соответствует вершина, а переходу – ребро. Нам необходимо несколько сузить поле деятельности, зафиксировав представление элементов и тип носителя, на котором мы планируем изображать диаграмму.

1.1. Выбор типа носителя

Диаграмма может быть изображена на плоскости, или, например, может быть построена объемная модель [6]. Для представления на мониторе современного персонального компьютера наиболее естественным представляется выбор плоского носителя.

1.2. Выбор представления элементов

Необходимо выбрать вид элементов графа. Будем изображать вершину как прямоугольник, а ребро – как ломаную линию с конечным числом изломов.

Выбор именно такого вида элементов объясняется исключительно субъективными эстетическими соображениями, а также сложившимися в области построения диаграмм традициями. Такой выбор диктует использование декартовой системы координат. Вершину принято изображать как прямоугольник со сторонами, ориентированными параллельно координатным осям. Для изображения вершины достаточно знать координаты левого верхнего угла, ширину и высоту, а для изображения ребра – координаты его начала, конца и всех точек излома.

1.3. Понятие укладки

Укладкой графа $G = (V; E)$ в декартовой системе координат $(X; Y)$ назовем множество $L = (G;\ F_V;\ F_E)$, где F_V – функция из множества вершин в множество параметров, необходимых для представления вершины в выбранной системе координат (в нашем случае $F_V : V \to X \times Y \times R \times R$, где последняя пара параметров – ширина и высота прямоугольника). F_E – функция из множества ребер в множество параметров, необходимых для представления ребра в выбранной системе координат (в нашем случае $F_E : V \to (X \times Y)^n, n \in N$, где параметр n – количество изломов – вообще говоря, меняется от ребра к ребру). Для простоты будем говорить, что в уложенном графе заданы геометрические параметры каждого элемента.

На задаче построения визуального изображения диаграммы по известным параметрам ее элементов в заданной системе координат (по заданной укладке) останавливаться здесь не будем.

Две укладки L, L' назовем **изоморфными**, если одна получается из другой путем параллельного переноса всех элементов на определенный вектор. Укладки на рис.1, например, являются различными. Далее мы, для простоты, будем говорить об изоморфных укладках как об одинаковых.

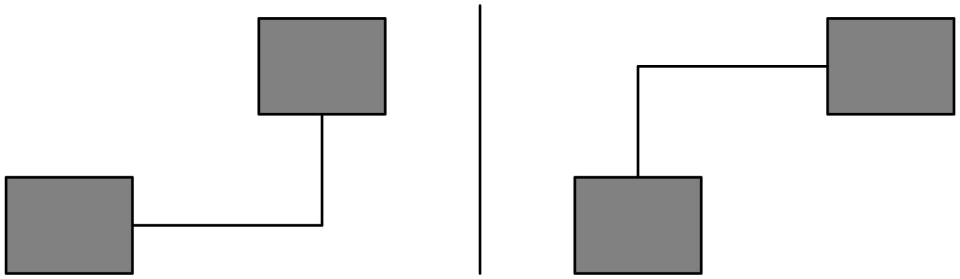

Рис. 1. Различные укладки

Таким образом, алгоритм укладывания диаграммы должен строить пару функций укладки $F_V;\ F_E$, точнее, соответствующие наборы геометрических параметров для каждой вершины.

Укладку на плоскости мы будем называть **плоской. Д**анный термин не является общепринятым, в литературе (например, [7]) чаще говорят о **плоском представлении** графа. Укладку, в которой ребра представляют собой ломаные, состоящие только из горизонтальных и вертикальных отрезков, будем называть **плоской ортогональной** (или просто **ортогональной**).

1.4. Оценка качества укладки

Теперь необходимо оценить качество укладки (точнее, качество изображения, полученного по некоторой укладке). Наиболее точный ответ способен дать человек. По-

пробуем выделить набор эстетик, по которым мы будем оценивать качество укладки. Основная задача – обеспечить «читаемость» графа (однозначность представления информации):

- количества пересечений ребер, прохождений ребер по вершинам и наложения вершин должны быть минимальны;
- площадь, занимаемая выпуклой оболочкой уложенного графа (или площадь минимального прямоугольника, включающего в себя уложенный граф), должна быть минимальной;
- доля свободного места на диаграмме не должны быть меньше некоторого предела;
- количество изломов и суммарная длина ребер должны быть минимальны.

К примеру, укладка слева на рис. 2 значительно лучше читается, чем укладка справа.

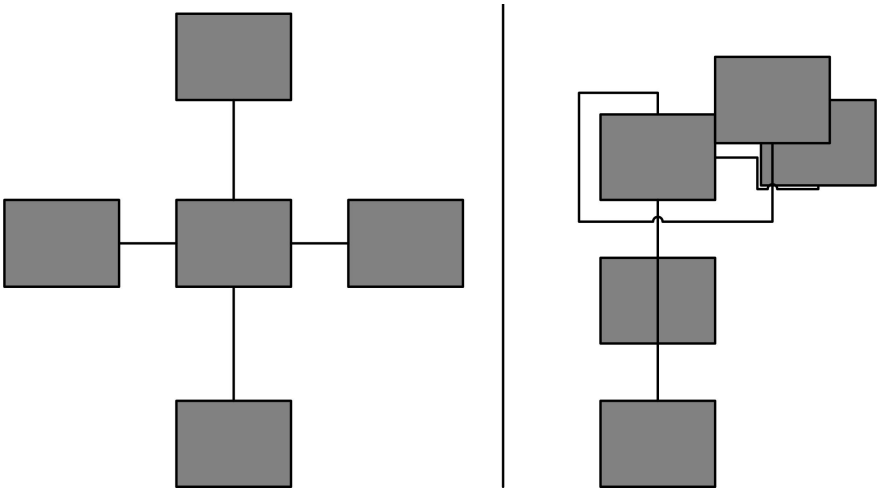

Рис. 2. Укладки графа

Граф, соответствующий диаграмме состояний, отличается от обычного графа возможностью вложения состояний друг в друга, наличием псевдосостояний, а также выделенных начальных и конечных состояний. Для оценки качества его укладки необходимо учесть дополнительные критерии:

- размер всех простых состояний (состояний, не имеющих вложенных состояний) необходимо максимально приблизить к заданному оптимальному размеру;
- расстояние между начальным и конечным состоянием должно быть достаточно велико.

Сравним две укладки на рис. 3. Укладка справа является предпочтительной, поскольку на ней автомат «визуально» выполняется слева направо (от начального состояния к конечному). Заметим, что приведенные эстетики не являются строгими и лишь дают понять, какими соображениями мы будем руководствоваться, оценивая результаты работы программ.

Обратим внимание на то, что в принятые в диаграмме состояний *UML* начальное и конечное состояние не имеют прямоугольной формы. Для простоты заменим их объемлющими прямоугольниками.

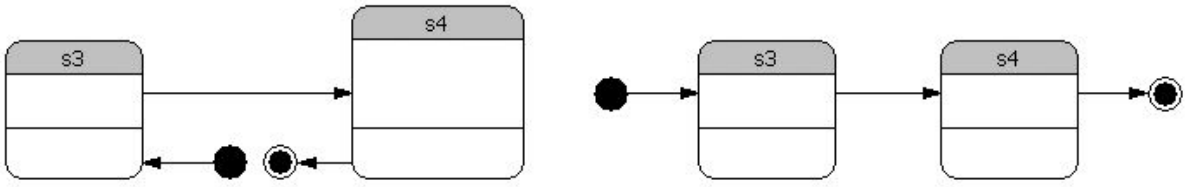

Рис. 3. Укладки диаграммы состояний

2. Типы алгоритмов укладки графов

Задача укладки графа не имеет и не может иметь универсального решения, в связи с тем, что набор критериев, применяемых для оценки качества укладки, зависит от типа диаграммы. В [8] приведена классификация алгоритмов, применяемых для решения данной задачи. Рассмотрим две большие группы, принципиально отличающиеся подходом к решению:

- алгоритмы с физическим аналогом [9];
- аналитические алгоритмы [7].

2.1. Алгоритмы с физическим аналогом

Эта группа алгоритмов ставит в соответствие графу некоторую физическую модель, например, систему пружин, которые стремятся сжаться до некоторой заданной длины (такие алгоритмы называют «пружинным методом») или систему стержней и шарниров, с вершинами – одноименными зарядами.

Для описания физической модели вводится понятие штрафной функции, задающей потенциальную энергию системы (такие алгоритмы еще называют методами минимизации потенциала). Задача укладывания преобразуется, таким образом, в задачу нахождения минимума этой функции, которая решается с помощью сдвига на случайный вектор каждой вершины графа и проверки изменения значения штрафной функции. Сдвиг вершин выполняется в цикле, условием выхода из которого является либо достижение локального минимума, либо достижение максимума числа допустимых итераций.

Наиболее популярная подгруппа группы алгоритмов с физическим аналогом – **методы отжига**. Они выделяются тем, что «колебания» системы затухают с каждой итерацией.

Для минимизации штрафной функции также можно использовать генетические алгоритмы. В [10] приводится обоснование такого подхода. Генетические алгоритмы строятся для минимизации тех же штрафных функций и отличаются, в основном, более высоким быстродействием.

Нахождение минимума потенциальной энергии также можно выполнить, решив систему дифференциальных уравнений, задающих ее. Для этого необходимо построить гладкую штрафную функцию, что является нетривиальной задачей. В [9] приведена ситуация, в которой малое изменение положения вершин должно приводить к резкому скачку штрафной функции.

На рис. 4 приведены две различные укладки одного и того же графа. Критерий «читаемости» графа, заключающийся в том, что пользователю должно быть понятно, какую пару вершин связывает каждое ребро, является для нас одним из важнейших. Следовательно, штрафная функция должна резко возрастать при нарушении условия читаемости. Теперь обратим внимание на то, что укладка слева на легко «читается», тогда как укладка справа «нечитаема» из-за наложения ребер. С другой стороны, координаты элементов обеих укладок отличаются незначительно. Это означает, что небольшой сдвиг вершин графа должен приводить к резкому скачку штрафной функции.

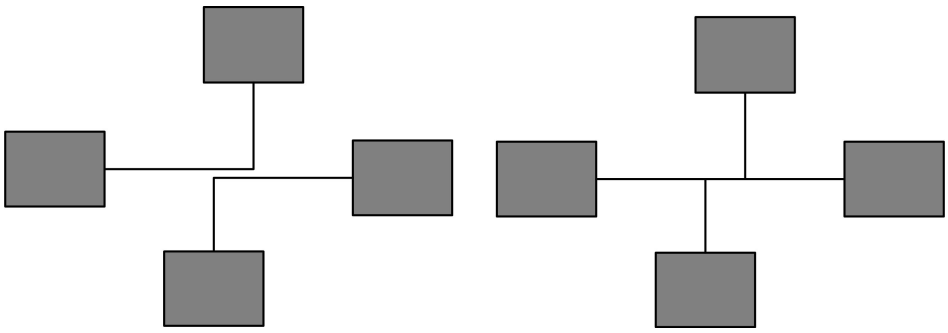

Рис. 4. Скачок штрафной функции

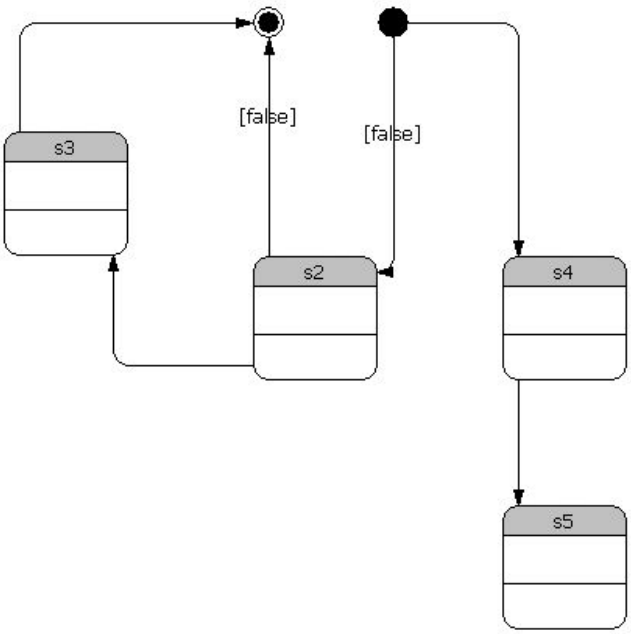

Рис. 5. Уложенная диаграмма

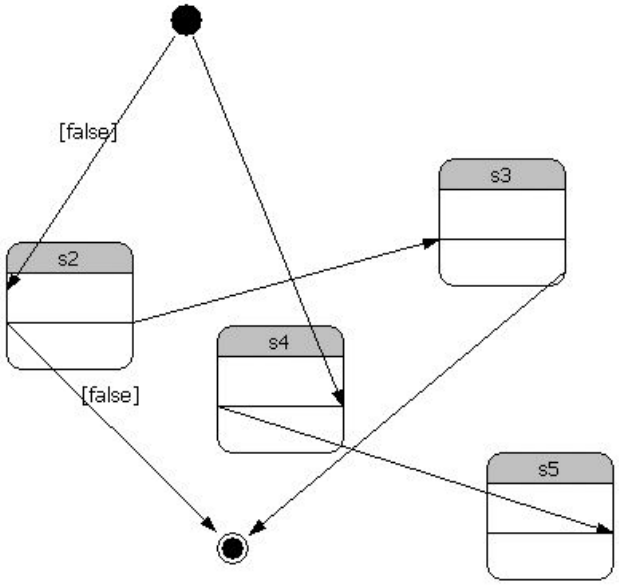

Рис. 6. Исходная диаграмма

Простота реализации метода отжига стала причиной реализации этого алгоритма для укладки диаграммы состояний в рамках проекта *UniMod*. Был выбран следующий подход: построим первоначальную укладку с помощью алгоритма отжига, а затем ортогонализуем ее (вообще говоря, с помощью аналитического алгоритма). Получившийся алгоритм можно назвать смешанным. При этом в процессе ортогонализации могут появиться дополнительные пересечения ребер, не учтенные алгоритмом отжига.

На рис. 5 приведена диаграмма состояний, полученная из диаграммы, приведенной на рис. 6, с помощью модифицированного алгоритма отжига.

2.2. Аналитические алгоритмы

Аналитические алгоритмы, в отличие от алгоритмов с физическим аналогом, представляют собой последовательность различных преобразований графа, приводящую к построению укладки. Это позволяет получать с их помощью гарантированный результат, удовлетворяющий выбранным критериям, так как критерии используются не для построения штрафной функции, а для построения самого алгоритма укладки.

К минусам аналитических алгоритмов можно отнести сложность их построения. Кроме того, они плохо поддаются модификации, и для постановки экспериментов приходится вносить серьезные изменения в сам алгоритм, а не в набор параметров, как это можно делать в случае использования, например, алгоритмов, базирующихся на штрафных функциях.

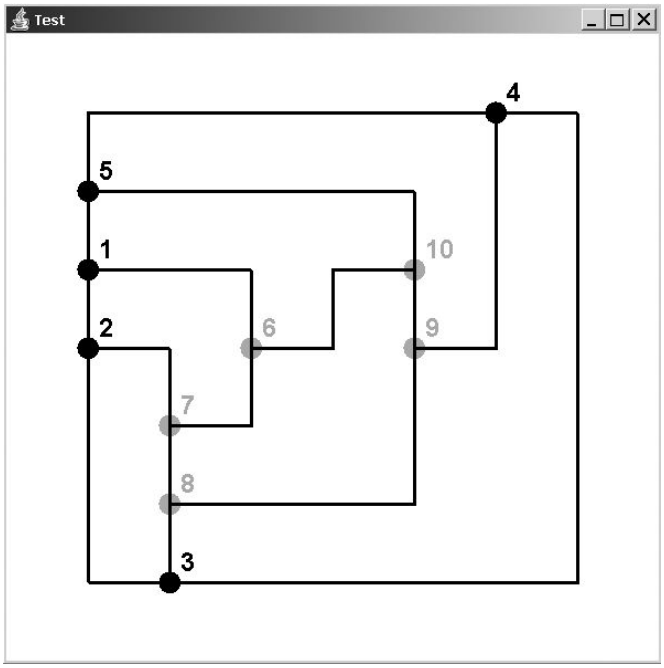

Рис. 7. Укладка графа

На практике (рассмотрено три реализации: [11–13]) аналитические алгоритмы позволяют относительно быстро получать наиболее качественные и красивые графы, в связи с этим для укладки диаграммы состояний в проекте *UniMod* было принято решение использовать эту группу алгоритмов в качестве основной.

Наиболее эффективным [14] в группе аналитических алгоритмов является алгоритм GIOTTO ([7, 15]). В [16] приводится обоснование применения модификации алгоритма GIOTTO для решения задачи укладки диаграммы состояний. Основная часть работы [16] посвящена укладке реберных меток, однако данная проблема этой в статье не затрагивается. В [7] изложены основы алгоритма GIOTTO. Было решено реализовать его и адаптировать к задаче укладки диаграммы состояний. Вторая задача на данный момент не решена окончательно. На рис. 7 приведен пример графа, уложенного с помощью модифицированного алгоритма GIOTTO.

Заключение

В настоящей работе исследована проблема плоской укладки диаграмм состояний *UML*. Поставлена задача укладки, дан обзор наиболее популярных методов.

На базе аналитического алгоритма плоской укладки графа GIOTTO, являющегося наиболее эффективным [14], разработан алгоритм, позволяющий уложить произвольную диаграмму состояний языка *UML*. Ценность алгоритма заключается в том, что он разработан специально для укладки *UML*-диаграммы состояний. Это позволило учесть специфические особенности диаграмм данного типа при их плоской укладке.

Кроме того, в работе описан смешанный алгоритм, сочетающий в себе простоту метода отжига и оригинальный аналитический алгоритм построения ортогональной укладки.

Литература

1. Гуров В.С., Мазин М.А. Веб-сайт проекта UniMod. http://unimod.sourceforge.net/
2. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. М.: ДМК. 2000.
3. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998.
4. Шалыто А.А., Туккель Н.И. Танки и автоматы //BYTE/Россия. 2003. № 2, с. 69–73. http://is.ifmo.ru/ (раздел «Статьи»).
5. Новиков Ф.А. Дискретная математика для программистов. СПб.: Питер. 2001.
6. Tamassia R. Advances in the Theory and Practice of Graph Drawing. http://www.cs.brown.edu/people/rt/papers/ordal96/ordal96.html
7. Battista G., Eades P., Tamassia R., Tollis I. Graph Drawing. Algorithms for the Visualization of Graphs. New Jersey: Prentice Hall. 1999.
8. Sugiyama K. Graph Drawing and Applications for Software and Knowledge Engineers. Singapore: Mainland Press. 2002.
9. Frutcherman T., Reingold E. Graph Drawing by Force-directed Placement, Software – Practice And Experience, 1991, Vol. 21, p1129–1164.
10. Makinen E., Seiranta M. Genetic algorithms for drawing bipartite graphs. International Journal of Computer Mathematics, 1994, Vol 53, No 3, p 157–166.
11. Веб-сайт проекта AGD (Algorithms for Graph Drawing). http://www.ads.tuwien.ac.at/AGD/
12. Веб-сайт проекта GDT (Graph Drawing Toolkit). http://www.dia.uniroma3.it/~gdt
13. Веб-сайт проекта Graph Layout Toolkit компании Tom Sawyer Software http://www.tomsawyer.com/tsl/tsl.java.php
14. Battista G., Garg A., Liotta G., Tamassia R., Tassinari E., Vargiu F., An experimental comparison of four graph drawing algorithms. Computational Geometry, 1997, 7(5-6), p303–325.
15. Tamassia R., Battista G., Batini C. Automatic graph drawing and readability of diagrams. IEEE Transactions on Systems Man Cybernetics, 1998, 18(1), p 61–79.
16. Klaw G., Mutzel P. Automatic layout and labeling of state diagrams. Materials of Graph Drawing conference, 1997.