УДК 681.3

Г. В. СОКОЛОВ
Пермский государственный университет

АНАЛИЗ АЛГОРИТМОВ АВТОМАТИЧЕСКОЙ УКЛАДКИ ГРАФОВ НА ПЛОСКОСТИ В РАМКАХ ЗАДАЧИ ВИЗУАЛИЗАЦИИ МОДЕЛЕЙ НА ГРАФАХ

Производится аналитический обзор существующих алгоритмов укладки графов на плоскости для их применения в унифицированном компоненте визуализации моделях на графах. При анализе алгоритмов особое внимание уделяется эстетическим критериям изображений графов на плоскости.

Графовое представление используется для отображения информации, допускающей моделирование в виде объектов и связей между ними. При визуализации моделях на графах возникают определенные трудности. Во-первых, не всегда удобно работать с визуальными моделями, отображаемыми в изначальном (исходном) виде (вследствие большого количества пересечений ребер затрудняется понимание семантики изучаемых моделей). Во-вторых, часто схемы каких-либо предметных областей описываются в текстовом виде, например на XML, где отсутствует информация о расположении вершин графовой модели. Поэтому задача автоматического расположения элементов графа в системах визуализации графов занимает одно из ключевых мест.

Автором разрабатывается унифицированный компонент визуализации графовых структур. Основными критериями, позволяющими наиболее наглядно визуализировать граф, являются:

– количество пересечений ребер произвольного графа должно быть минимальным (в идеальном случае – граф планарный, нулевое количество пересечений ребер; в случае если граф непланарный, то необходимо выделить максимальный планарный подграф);

– суммарная площадь, занимаемая графом (площадь минимального выпуклого многоугольника, покрывающего изображение графа), должна быть минимальна, но доля свободного места не должна быть меньше некоторого предела;

– количество наложений вершин друг на друга должно быть минимальным.

Кроме того, естественно, алгоритм автоматической укладки графа на плоскости должен быть эффективным по времени исполнения (полиномиальной сложности).

Таким образом, разработчик должен решить задачу многокритериальной оптимизации.

Классификация алгоритмов укладки графа выглядит следующим образом [4]:

– алгоритмы на основе физической модели,

– аналитические алгоритмы,

– генетические алгоритмы.

Алгоритмы на основе физической модели. Данная группа алгоритмов ставит в соответствие графу некоторую физическую модель. Вводят так называемую штрафную функцию, которую минимизируют. Выделяют «пружинный метод» и метод отжига.

В «пружинном методе» [3] граф интерпретируется как некоторая упругая механическая система, в вершинах которой находятся точки системы, а ребра соответствуют упругим силовым связям («пружинкам»). Штрафная функция – потенциальная энергия системы, а ее минимуму соответствует условие равновесия системы.

Если силовые связи подчиняются закону Гука, то сила натяжения пружины пропорциональна ее длине. В этом случае потенциальная энергия системы представляет собой положительную функцию

$$W = \frac{1}{2} \sum_{(p_i, p_j) \in E} w_{ij} d^2(p_i, p_j),$$

где E – множество ребер графа; d^2 (p_i, p_j) – квадрат расстояния между вершинами графа p_i, p_j, декартовы координаты которых $(x_i\ y_i)$ и (x_j, y_j); $d^2(p_i, p_j) = (x_i - x_j)^2 + (y_i - y_j)^2$; w_{ij} – произвольные положительные веса этих ребер (в частности, можно взять все веса равными 1).

Функция W зависит от координат вершин, не принадлежащих внешней грани: $W(x_i, y_i)$.

Далее для нахождения минимума используется условие равенства первых производных нулю, т.е. $\frac{\partial W}{\partial x_i} = 0; \frac{\partial W}{\partial y_i} = 0;$

После несложных преобразований:

$$x_i \sum_{p_j \in V(p_i)} w_{ij} - \sum_{p_{j \in V^{\text{in}}(p_i)}} w_{ij} x_j = \sum_{p_j \in V^{\text{out}}(p_i)} w_{ij} x_j,$$

$$y_i \sum_{p_j \in V(p_i)} w_{ij} - \sum_{p_{j \in V^{\text{in}}(p_i)}} w_{ij} y_j = \sum_{p_j \in V^{\text{out}}(p_i)} w_{ij} y_j,$$

где $V(p_i)$ – окрестность вершины p_i, т.е. множество смежных с p_i вершин;$V^{in}(p_i)$ – множество вершин из $V(p_i)$, не принадлежащих внешней грани; $V^{out}(p_i)$ – множество вершин из $V(p_i)$, принадлежащих внешней грани.

Нетрудно заметить, что данный алгоритм имеет полиномиальную сложность.

Можно выделить следующие недостатки «пружинного метода»:

– необходимо выделять внешнюю грань; алгоритм позволяет ответить на вопрос о планарности графа, но не позволяет выделить максимальный планарный подграф, что не позволяет работать с произвольными графами;

– алгоритм не дает возможность учитывать размеры вершин графа, что не удовлетворяет поставленному вначале третьему требованию;

– существует привязка вершин графа к координатам на плоскости, которая не дает возможности использовать эффективные методы визуализации графа с минимальной площадью размещения.

Модификацией данного алгоритма является нелинейный силовой метод [3]. Укладка связного графа обеспечивается уравновешиванием сил притяжения и отталкивания, зависящих от расстояния между вершинами d: для всех пар вершин действует сила отталкивания $R(d)$, монотонно убывающая с ростом d, для всех пар смежных вершин действует сила притяжения $A(d)$, монотонно возрастающая с ростом d. В этом случае нивелируется первый недостаток предыдущего метода, но по-прежнему остается нерешенной вторая и третья проблема. Более того, минимизация штрафной функции сводится к решению нелинейной системы дифференциальных уравнений, что приводит к дополнительным вычислительным расходам, а также накладывает ограничения на начальные условия (например, при решении системы методом градиентного спуска).

Следующий метод на основе физической модели – метод отжига.

Это техника оптимизации, использующая упорядоченный случайный поиск на основе аналогии с процессом образования веществом кристаллической структуры с минимальной энергией при охлаждении [8].

Идея данного метода состоит в следующем: граф представляет собой некоторую систему. Вводится понятие «энергетического уровня» системы, т.е. каждому состоянию системы, а именно раскладке графа, соответствует энергия E, пропорциональная, например, количеству пересечений ребер графа. Пусть $E = f(x), x \in S$, где S – множество различных укладок графа. Теперь необходимо минимизировать E.

Таким образом, минимуму энергии будет соответствовать укладка графа с минимальным количеством пересечений ребер.

В каждый момент времени предполагается заданной температура системы, которая с течением времени охлаждается. Каждое следующее состояние выбирается в соответствии с заданным вероятностным семейством распределений $G(x,T)$. После генерации нового состояния $x^l = G(x,T)$ система с вероятностью $H(\Delta E,T)$ (вероятность принятия нового состояния) переходит к следующему шагу в состояние x^l, иначе процесс генерации x^l повторяется. $\Delta E = f(x^l) - f(x)$. В качестве $H(\Delta E,T)$ обычно берут следующую функцию: $H(\Delta E,T) = (-\Delta E / T)$. При ее использовании $H(\Delta E,T)$ больше единицы в случае $\Delta E < 0$, тогда соответствующая вероятность считается равной единице. Таким образом, если новое состояние дает лучшее значение оптимизируемой функции, то переход в это состояние произойдет в любом случае.

Итак, конкретная схема отжига определяется следующими параметрами:

– выбором закона распределения температуры от шага $T(k)$;

– выбором порождающего семейства распределений $G(x,T)$;

– выбором функции вероятности принятия нового состояния $H(\Delta E,T)$.

Существуют более эффективные схемы отжига, например сверхбыстрый отжиг [8], который позволяет довольно быстро искать глобальный минимум.

Достоинства метода:

– благодаря универсальности подхода, используемого методом отжига, а именно свободе выбора функции энергии системы E, можно оптимизировать любые требуемые критерии, в том числе и площадь, занимаемую укладкой графа;

– алгоритм позволяет не только ответить на вопрос о планарности графа, но и выделить максимальный планарный подграф.

Недостатки:

– требует длительной настройки параметров для конкретной задачи;

– эффективно работает только для небольших графов (в связи с трудоемкостью метода);

– не учитывается размер вершин графа;

– сложно подобрать функцию E для быстрой сходимости данного метода.

Аналитические алгоритмы. На практике аналитические алгоритмы представляют собой последовательность преобразований графа, приводящих к укладке графа. Главное преимущество данной группы алгоритмов состоит в том, что с их помощью получается гарантированный результат, в отличие от алгоритмов на базе физической модели. К данной группе алгоритмов можно отнести гамма-алгоритм, алгоритм GIOTTO, а также другие алгоритмы [7], не рассматриваемые в данной работе.

Гамма-алгоритм основан на выделении сегментов в графе и их определенной укладке в нужным образом выбранных гранях графа [2]. Достоинством данного алгоритма является возможность производить укладку графа с криволинейными ребрами. Но главным недостатком данного алгоритма является то, что он лишь позволяет укладывать планарные графы, что противоречит первому требованию, поставленному в начале данной работы. Более того, данный алгоритм плохо поддается модификации.

Наиболее эффективным является алгоритм GIOTTO [6]. Основные шаги данного алгоритма:

1. Преобразование графа в связный;
2. Преобразование графа в двусвязный;
3. Выделение планарного подграфа;
4. Построение *st*-графа;
5. Построение планарного «надграфа»;
6. Разбиение вершин инцидентностью больше четырех на цепочки;
7. Построение ортогонального представления;
8. Минимизация площади, занимаемой уложенным графом;
9. Удаление фиктивных элементов, созданных в процессе предыдущих шагов;
10. Восстановление информации об ориентации ребер (информация теряется на четвертом шаге).

Таким образом, алгоритм GIOTTO позволяет выделить максимальный планарный подграф, а также минимизировать площадь, занимаемую уложенным графом. Минус данного алгоритма состоит в том, что его сложно модифицировать, а следовательно, практически невозможно учесть третье требование об учете размеров вершин графа.

Генетические алгоритмы (ГА). Данные алгоритмы, подобно методу отжига являются универсальными с точки зрения решения оптимизационных задач [5] и имеют аналогию с природным механизмом.

Все внешние данные особи кодируются ее цепью ДНК (генотипом). Отдельные участки этой цепи (гены) определяют различные па-

раметры особи. Согласно теории эволюции Чарльза Дарвина, особи популяции конкурируют между собой за пищу и за привлечение брачного партнера. Те особи, которые наиболее приспособлены к окружающим условиям, проживут дольше и создадут более многочисленное потомство, чем их собратья. Скрещиваясь, они будут передавать потомкам часть своего генотипа. Некоторые дети совместят в себе части цепи ДНК, отвечающие за наиболее удачные качества родителей и, таким образом, окажутся еще более приспособленными.

Те особи, которые не обладают качествами, способствующими их выживанию, с большой вероятностью не проживут долго и не смогут создать многочисленное потомство. Кроме того, им сложнее будет найти хорошую пару для скрещивания, поэтому с большой вероятностью генотип таких особей исчезнет из генофонда популяции.

Изредка происходит мутация: некоторый случайный нуклеотид цепи ДНК особи может измениться на другой. Если полученная цепь будет использоваться для создания потомства, то у детей возможно появление совершенно новых качеств.

Естественный отбор, скрещивание и мутация обеспечивают развитие популяции. Каждое новое поколение в среднем более приспособлено, чем предыдущее, т.е. оно лучше удовлетворяет требованиям внешней среды.

Теперь, пусть необходимо найти максимум (минимум) функции $f(x_1, x_2, ..., x_n)$. Эта функция называется *функцией приспособленности* (*fitness function*) и используется для вычисления приспособленности особей. Она должна принимать неотрицательные значения, а ее область определения должна быть ограниченным множеством. В случае задачи планаризации графа функцией приспособленности может выступать, например, как самый тривиальный вариант, функция количества пересечений ребер графа (требуется найти ее минимум).

Каждый параметр функции приспособленности будет кодироваться битовой строкой. *Особью* [5] будет называться строка, являющаяся конкатенацией строк всего упорядоченного набора параметров:

101100 11001011 01101100 1100 1 11101

| x_1 | x_2 | | | x_n |

Приспособленность особи высчитывается следующим образом: строка разбивается на подстроки, кодирующие параметры. Затем для каждой подстроки высчитывается соответствующее ей значение параметра, после чего приспособленность особи вычисляется как значение функции приспособленности от полученного набора.

Вообще, от конкретной задачи зависят только такие параметры генетического алгоритма, как функция приспособленности и кодирование решений. Остальные шаги для всех задач производятся одинаково, в этом проявляется универсальность алгоритма.

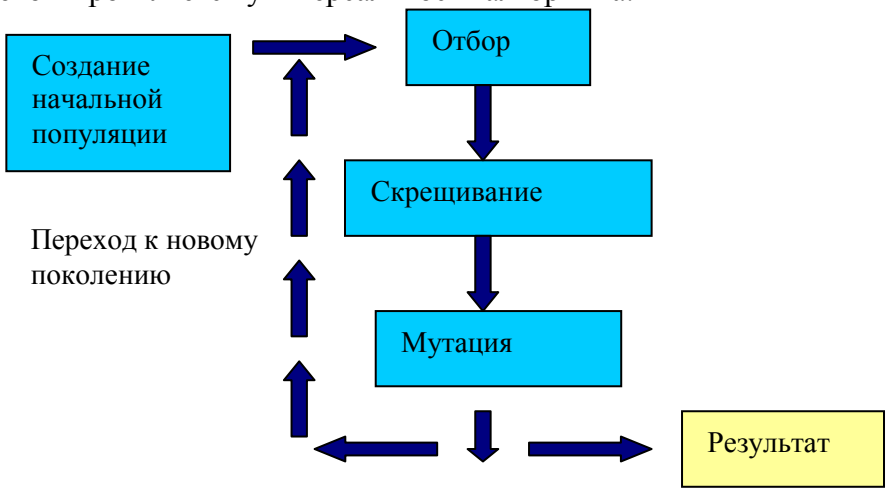

Рис. Основные этапы генетического алгоритма

В классическом гинетическом алгоритме [5] начальная популяция формируется случайным образом. Фиксируется размер популяции (количество особей N), который не изменяется в течение работы всего алгоритма. Каждая особь генерируется как случайная L-битная строка, где L – длина кодировки особи, она тоже фиксирована и для всех особей одинакова.

Следует заметить, что каждая особь является одним из решений поставленной задачи. Более приспособленные особи – это более подходящие решения. Этим генетический алгоритм отличается от большинства других алгоритмов оптимизации, которые оперируют лишь с одним решением, улучшая его.

Шаг алгоритма состоит из трех стадий (рисунок): генерация промежуточной популяции (*intermediate generation*) путем отбора (*selection*) текущего поколения (*current generation*), скрещивание (*recombination*) особей промежуточной популяции путем *кроссовера* (*crossover*), что приводит к формированию нового поколения (*next generation*), и мутация нового поколения.

Промежуточная популяция – это набор особей, которые получили право размножаться. Приспособленные особи могут быть записаны туда несколько раз. «Плохие» особи с большой вероятностью туда вообще не попадут.

Вероятность, что особь попадет в промежуточную популяцию, пропорциональна ее приспособленности, т.е. работает пропорциональный отбор.

После отбора особи промежуточной популяции случайным образом разбиваются на пары. Каждая из них с вероятностью p_c скрещивается, т.е. к ней применяется оператор кроссовера, в результате чего получаются два потомка. Они записываются в новое поколение. Если же паре не выпало скрещиваться, в новое поколение записываются сами особи этой пары.

В классическом гинетическом алгоритме [5] применяется *одноточечный оператор кроссовера* (*1-point crossover*): для родительских хромосом случайным образом выбирается точка раздела, и они обмениваются отсеченными частями. Полученные две строки являются потомками:

11010 01100101101 $\Rightarrow$ 10110 **01100101101**

10110 10011101001 $\Rightarrow$ **11010** 10011101001

К полученному в результате скрещивания новому поколению применяется *оператор мутации*. Каждый бит каждой особи популяции с вероятностью p_m инвертируется. Эта вероятность обычно очень мала, менее 1 %.

10110011**0**0101101 $\Rightarrow$ 10110011**1**0101101

Таким образом, процесс отбора, скрещивания и мутации приводит к формированию нового поколения. Шаг алгоритма завершается объявлением нового поколения текущим. Далее все действия повторяются.

Вообще говоря, такой процесс эволюции может продолжаться до бесконечности. Критерием останова может служить либо заданное количество поколений, либо достижение *состояния схождения* (*convergence*) популяции.

Состоянием схождения называется такое состояние популяции, когда все особи популяции почти одинаковы (различаются в несколько нуклеотидов) и находятся в области некоторого экстремума. В такой ситуации кроссовер практически никак не изменяет популяции. А вышедшие из этой области за счет мутации особи склонны вымирать, т.к. чаще имеют меньшую приспособленность, особенно если данный экстремум является глобальным максимумом.

В [1] разработан генетический алгоритм, выполняющий планарную укладку графа за полиноминальное время. Для определения планарности графа используется критерий Мак-Лейна, сформулированный в терминах базиса циклов. Использован оригинальный способ ко-

дирования решений. Так, каждая хромосома имеет p+3 генов (разрядов): p разрядов – это комбинация циклов, представляющая собой вариант плоской укладки графа, последние три разряда несут информацию о том, какие ребра и сколько раз содержатся в выбранной комбинации (0, 1 или 2 раза соответственно для (p+1), (p+2), (p+3)-генов). В качестве функции приспособленности промежуточных решений используется аддитивная функция двух переменных $f'(H) = (\delta'_m, N'_c)$, где δ'_m – относительное увеличение числа ребер, N'_c – относительное увеличение числа циклов. Количественное значение функции приспособленности рассчитывается из числа совпадений ребер в соответствующих разрядах анализируемых решений, а также числа циклов, которые добавляются (удаляются) в базовом решении. В качестве оператора кроссинговера используется двухточечный суммирующий кроссинговер.

Следует выделить следующие преимущества генетических алгоритмов при решении задачи планаризации графов:

– возможность применения к произвольным графам,

– полиномиальная сложность,

– выделение максимального планарного подграфа,

– нет привязки к координатам вершин графа на плоскости, что позволит использовать эффективные методы визуализации графа, например для минимизации площади размещения.

Недостатки:

– проблема кодирования решения,

– проблема выбора эффективной функции приспособленности.

Среди рассмотренных в данной работе алгоритмов нет таких, которые осуществляют многофакторную оптимизацию по всем требуемым критериям.

Но наиболее подходящим для реализации в унифицированном компоненте визуализации графовых структур является использование генетического алгоритма в качестве алгоритма планарной укладки, т.к. данный алгоритм позволяет не только определить, планарен ли граф, но и выделить максимальный планарный подграф в непланарном графе. Более того, алгоритм гибок, легко поддается модификации, нет привязки к координатам вершин на плоскости, так что можно использовать эффективные методы поуровневой визуализации [3] с учетом размера вершин графа, а также минимизации площади, занимаемой графом. Кроме того, генетический алгоритм можно будет модифицировать, используя островную модель [5].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гладков Л.А. Решение задачи планаризации графов на основе бионических технологий // Вестник ЮНЦ РАН. – Т. 1. – Вып. 2.– 2005.
2. Иринев А., Каширин В. Алгоритм плоской укладки графов. – URL: rain.ifmo.ru/cat/view.php/theory/graph-coloring-layout/layout-2006.
3. Касьянов В.Н., Евстигнеев В.А. Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ-Петербург, 2003. – 1104 с.
4. Коротков М.А. Разработка и реализация алгоритма укладки диаграмм состояний. – URL: rain.ifmo.ru/cat/view.php/theory/graph-coloring-layout/uml-layout-2005.
5. Яминов Б. Генетические алгоритмы. – URL: rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005.
6. Graph Drawing. Algorithms for the Visualization of Graphs / G. Battista [et al.].– New Jersey: Prentice Hall, 1999.
7. Sigiyama K. Graph Drawing and Applications for Software and Knowledge Engineers. – Singapore: Mainland Press, 2003. – 218 p.
8. Лопатин А. Метод отжига. – URL:cs-seminar. spb.ru/reports/52. pdf.