МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Планарные графы

АВТОРЕФЕРАТ

дипломной работы

студента 6 курса 631 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Асеева Никиты Андреевича

Научный руководитель

зав. кафедрой, д. ф.-м. н., доцент ___________ М. Б. Абросимов

23.01.2021 г.

Заведующий кафедрой

д. ф.-м. н., доцент ___________ М. Б. Абросимов

23.01.2021 г.

Саратов 2021

ВВЕДЕНИЕ

В 1736 году математик Леонард Эйлер опубликовал статью о решении задачи кёнигсбергских мостов. Можно без преувеличения сказать, что так было положено начало новой науке – теории графов. И хотя сперва теория графов ассоциировалась с занимательными задачами и загадками, с течением времени ей стали находиться все более разнообразные применения. Сегодня графы повсюду: в картографии, теории сетей, навигации, схемотехнике и других областях.

В данной работе будут рассмотрены планарные графы, основные понятия, сведения и теоремы, связанные с ними, будет рассмотрен Гамма-алгоритм для проверки планарности, алгоритм Фрейззе – Паха – Поллака для построения плоской укладки с прямыми рёбрами, а также алгоритм генерации случайных планарных графов. Планарные графы находят применение в системах с графическим паролем, проектировании компьютерных сетей, создании микросхем.

В практической части будет представлена программа, позволяющая проверить граф на планарность, сгенерировать заданное количество планарных двусвязных графов с определенным числом вершин, визуализировать граф путём силовой укладки, если граф непланарен, или путём алгоритма Фрейззе – Паха – Поллака, предназначенного для построения прямолинейных изображений.

Дипломная работа состоит из введения, 7 разделов, заключения, списка использованных источников и приложений. Общий объём работы составляет 75 страниц, из них 49 страниц – основное содержание, включая 53 рисунка и 2 таблицы, список использованных источников из 15 наименований.

КРАТКОЕ СОДЕРЖАНИЕ

Основные понятия и определения

Неориентированным графом (или, для краткости, *графом*) называется пара G = (V, α), где α – симметричное и антирефлексивное отношение на множестве вершин V. *Изображением* графа G назовем отображение каждой вершины этого графа на некоторую точку $P(v) = (v_x, v_y)$ плоскости и каждого ребра $\{u,\ v\}$ на некоторую Жорданову кривую с концами $P(u)$ и $P(v)$. Изображение называется плоским, если два ребра пересекаются разве что только своими концами. Две планарные укладки эквивалентны, если для каждой вершины v они имеют одинаковые последовательности ребёр, когда вершина v обходится по часовой стрелке. Классы эквивалентности изображений назовем *укладками*. Также стоит сказать, что *решётчатым изображением* называется отображение f: $v \in V \to (v_x, v_y) \in \mathbb{Z}^2$. Граф называется *планарным*, если он допускает плоское изображение. *Максимальным планарным графом* называется граф, который при добавлении любого ребра перестает быть планарным.

Гамма-алгоритм

Для проверки планарности будем использовать Гамма-алгоритм. Алгоритм γ укладки графа G представляет собой процесс последовательного присоединения к некоторому уложенному подграфу $\bar{G}$ графа G новой цепи, оба конца которой принадлежат $\bar{G}$. Тем самым эта цепь разбивает одну из граней графа $\bar{G}$ на две. При этом в качестве начального плоского графа $\bar{G}$ выбирается любой простой цикл графа G. Процесс продолжается до тех пор, пока не будет построен плоский граф, изоморфный графу G, или присоединение некоторой цепи окажется невозможным. В последнем случае граф G не является планарным. Пусть построена некоторая укладка подграфа $\bar{G} = (\bar{V}, \bar{\alpha})$ графа G = (V, α). *Сегментом* S относительно $\bar{G}$ будем называть подграф графа G одного из следующих двух видов:

1. ребро $e = uv \in \alpha$ такое, что $e \notin \bar{\alpha}$, $u, v \in \bar{V}$;

2. связную компоненту графа $G - \bar{G}$, дополненную всеми ребрами графа G, инцидентными вершинам взятой компоненты, и концами этих ребер.

Вершину v сегмента S относительно $\bar{G}$ будем называть *контактной*, если $v \in \bar{V}$. *Допустимой гранью* для сегмента S относительно $\bar{G}$ называется грань Г графа $\bar{G}$, содержащая все контактные вершины сегмента S. Через Г(S) будем обозначать множество допустимых граней для S. Может оказаться, что Г(S) = ∅. Простую цепь L сегмента S, соединяющую две различные контактные вершины и не содержащую других контактных вершин, назовем *α-цепью*. Очевидно, что всякая α-цепь, принадлежащая сегменту, может быть уложена в любую грань, допустимую для этого сегмента.

Теперь формально опишем алгоритм γ:

1. Выберем некоторый простой цикл C графа G и уложим его на плоскости; положим $\bar{G} = C$.
2. Найдем грани графа G и сегменты относительно $\bar{G}$. Если множество сегментов пусто, то перейдем к шагу 8.
3. Для каждого сегмента S определим множество Г(S).
4. Если существует сегмент S, для которого Г(S) = ∅, то граф G непланарен. Конец. Иначе перейдем к шагу 5.
5. Если существует сегмент S, для которого имеется единственная допустимая грань Г, то перейдем к шагу 7. Иначе – к шагу 6.
6. Для некоторого сегмента S (Г(S) > 1) выбираем произвольную допустимую грань Г.
7. Поместим произвольную α-цепь $L \in S$ в грань Г; заменим $\bar{G}$ на $\bar{G} \cup L$ и перейдем к шагу 2.
8. Построена укладка $\bar{G}$ графа G на плоскости. Конец.

Генерация планарных графов

Генерация случайного планарного графа с заданным числом вершин – задача не тривиальная. Для генерации планарных графов будем использовать *генераторы Больцмана* – эффективный способ генерации комбинаторных объектов. Необходимо сказать, полученный граф двусвязен, а также *размеченный,* то есть вершинам присваиваются различные метки, и *простой,* то есть без петель и кратных рёбер. Я остановился на этом классе, поскольку во время работы многих алгоритмов по проверке планарности и укладке граф так или иначе декомпозируется на блоки, которые затем укладываются по отдельности.

Чтобы понять, как работает генератор, нужно дать основные определения из области аналитической комбинаторики. *Комбинаторным классом* назовем счетное множество математических объектов вмести с функцией размера, которая отображает каждый объект в ненулевое число. Объект может состоять из *размеченных атомов* (*L-атомы*), то есть составных частей, которым присваиваются различные метки, и *неразмеченных атомов* (*U-атомы*). Тогда говорят, что класс C – *смешанный*. Для $\gamma \in C$ мы обозначим $|\gamma|$ число размеченных атомов и $\|\gamma\|$ – число неразмеченных атомов. Также с комбинаторными классами связаны их производящие функции. Для комбинаторных классов производящие функции будут принимать следующую форму, для классов с размеченными атомами и смешанных классов соответственно: $C(x) := \sum_{\gamma \in C} \frac{x^{|\gamma|}}{|\gamma|!}$ и $C(x, y) := \sum_{\gamma \in C} \frac{x^{|\gamma|}}{|\gamma|!} y^{\|\gamma\|}$.

Определим генератор Больцмана – эффективный метод генерации объектов любых декомпозируемых классов C. Вместо того, чтобы фиксировать размер объекта, мы позволяем рисовать объекты случайно их класса, но с некоторой вероятностью. Пусть дано x, тогда распределение Больцмана присваивает каждому объекту из C следующую вероятность $P_x(\gamma) = \frac{x^{|\gamma|}}{|\gamma|! C(x)}$. Заметим, что распределение равномерно, то есть два объекта с одинаковым весом рисуются с одинаковой вероятностью. Генератор Больцмана для класса с размеченными атомами C – это процедура $\Gamma C(x)$, которая для каждого допустимого x рисует объект из C случайно с вероятностью P_x. Для смешанных классов вероятность будет иметь вид $P_{x,y}(\gamma) = \frac{1}{C(x,y)} \frac{x^{|\gamma|}}{|\gamma|!} y^{\|\gamma\|}$. Равномерность распределения сохраняется, генератор определяется аналогично. Определим базовые генераторы и конструкции, позволяющие сделать генераторы для других классов, у которых производящие функции будут получаться по правилами аналитической комбинаторики. Для вывода правил понадобится вспомнить два распределения:

- Случайная величина имеет распределение Бернулли с параметром $p \in (0, 1)$, если она равна 1 с вероятностью p и равна 0 с вероятностью $1 - p$.
- Даны $\lambda > 0$ и $d > 0$, целое, случайная величина имеет распределение Пуассона с условием (conditioned), если она определяется как: $P(k) = \frac{1}{exp_{\geq d}(\lambda)} \frac{\lambda^k}{k!}$, где $exp_{\geq d}(z) := \sum_{k \geq d} \frac{z^k}{k!}$.

Таблица 1 – Генераторы для базовых классов

Базовый класс	Генератор Больцмана
1-класс	Вернуть 0-атом
L-класс	Вернуть L-атом
U-класс	Вернуть U-атом

Таблица 2 – Генераторы для операций

Операция	Генератор Больцмана
$C = A + B$	$\Gamma C(x, y)$: если Bern $\left(\frac{A(x,y)}{C(x,y)}\right)$ вернуть $\Gamma A(x, y)$, иначе вернуть $\Gamma B(x, y)$;
$C = A * B$	$\Gamma C(x, y)$: γ ← $(\Gamma A(x, y), \Gamma B(x, y))$ {независимые вызовы}; distributeLabels(γ); вернуть γ;
$C = SET_{\geq \mathrm{d}}(B)$	$\Gamma C(x, y)$: k ← $Pois_{\geq \mathrm{d}}(B(x, y))$; γ ← $(\Gamma B(x, y), \dots, \Gamma B(x, y))$ {k независимых вызовов}; distributeLabels(γ); вернуть γ;

$C = A \circ_L B$	$\Gamma C(x,y)$: $\gamma \leftarrow \Gamma A(B(x,y),y)$; Для каждого L-атома v из γ сделать: заменить v на $\gamma_v \leftarrow \Gamma B(x,y)$ {независимые вызовы}; distributeLabels(γ); вернуть γ;
$C = A \circ_U B$	$\Gamma C(x,y)$: $\gamma \leftarrow \Gamma A(x, B(x,y))$; Для каждого U-атома e из γ сделать: заменить e на $\gamma_e \leftarrow \Gamma B(x,y)$ {независимые вызовы}; distributeLabels(γ); вернуть γ;

Алгоритм начинается с генерации 3-связных планарных графов, которые хорошо трактуются с точки зрения комбинаторики. По теореме Уитни, каждый 3-связный планарный граф имеет уникальную укладку на плоскости (в точности до отражения), поэтому они эквивалентны 3-связным картам – плоским укладкам. 3-связные карты с корнем – класс, получаемый из исходного путем пропуска двух размеченных атомов и одного неразмеченного при подсчёте атомов – эквивалентны классу неприводимых квадрангуляций с корнем, которые являются картами со всеми гранями в виде четырёхугольников и где любой 4-цикл – это контур грани. Класс неприводимых квадрангуляций с корнем эквивалентен классу допустимых асимметричных двуцветных неприводимых рассечений гексагона с корнем – в отличие от предыдущего класса внешняя грань становится шестиугольником, и нет внутреннего пути длины 3. Рассечения из предыдущего предложения без корня эквивалентны классу двуцветных двоичных асимметричных деревьев без корня, чьи смежные узлы можно раскрасить в разные цвета. Таким образом, генерируя деревья, сможем получить через цепочку преобразований и операций с генераторами генератор для 3-связного планарного графа. Следующим шагом в построении генератора для 2-связного планарного графа будет использование декомпозиции через сети, которая гарантирует, что 2-связный граф (с корнем) может быть собран из 3-связных графов (с корнем). Эта декомпозиция имеет дело с так называемыми сетями, где сеть обозначает связанный граф N с двумя отличными вершинами 0 и ∞, называемыми полюсами, такими, что граф N*, полученный добавлением ребра между 0 и ∞, является 2-связным. Заметим, что сети во многом совпадают с 2-связными графами с корнем, однако не изоморфны. Сеть, сделанная генератором $\Gamma D(z, y)$, будет состоять из последовательно-параллельного остова и коллекции 3-связных планарных графов, которые прикреплены к ребрам остова; финальный двусвязный планарный граф будет получаться путём добавления ребра в сеть, связывая тем самым полюса.

Визуализация планарных графов

Для визуализации планарных графов будем пользоваться алгоритмом Фрейззе – Паха – Поллака. Пусть G есть максимальный планарный плоский граф с n вершинами, пусть u_0, u_1. u_2 являются *внешними вершинами* – то есть лежащими на внешней грани – в порядке против часовой стрелки. *Каноническим порядком* G назовём такой порядок вершин v_1, ..., v_n графа, что для него выполняется следующие условия:

1. $v_1 = u_1$, $v_2 = u_2$.
2. Для $3 \leq k \leq n$ пусть G_k будет плоский подграф графа G, порожденный вершинами v_1, ..., v_k и пусть C_k будет внешней гранью G_k. Вершина v_k находится на C_k. Также, для $k < n$ вершина v_k имеет хотя бы одного соседа в $G – G_k$.
3. Для каждого $3 \leq k \leq n – 1$ подграф G_k есть двусвязный и *внутренне максимальный* (все внутренние грани являются треугольниками).
4. $v_n = u_0$.

На шаге k ($3 \leq k \leq n$) вершина v_k и рёбра между v_k и ее соседями в $C_{k–1}$ добавляются в текущий граф $G_{k–1}$. Для каждого $3 \leq k \leq n$ мы обозначим последовательность $v_1 = w_1$, w_2, ..., $w_t = v_2$ как последовательность вершин $C_{k–1}$, когда проходим по часовой стрелке. Для каждого ($3 \leq k \leq n$) обозначим w_p, ..., w_q как последовательность вершин из $C_{k–1}$, которые смежны с v_k. После того, как v_k была добавлена в $G_{k–1}$, вершины w_{p+1}, ..., $w_{q–1}$ перестают быть внешними; v_k *покрывает* эти вершины.

Алгоритм Фрейззе – Паха – Поллака состоит в конструировании плоского изображения с прямыми рёбрами n-вершинного максимального плоского графа на сетке размера $(2n – 4) \times (n – 2)$. Общий план алгоритма такой:

1. Вершины помещаются на сетку одна за раз, следуя каноническому порядку.
2. На каждом шаге контур изображения текущего графа сохраняет некоторые инварианты, включая ограничения на наклон ребёр.
3. Когда вершина помещается на сетку, некоторые ранее помещённые вершины сдвигаются налево и некоторые – направо, чтобы сохранить инварианты и плоское свойство.

Стоит более подробно описать алгоритм. Пусть G – n-вершинный максимальный плоский граф, и $v_1, ..., v_n$ есть канонический порядок вершин G. Обозначим $P(v) = (x(v), y(v))$ как текущую позицию v на сетке. Для каждой v, мы поддерживаем множество вершин, которые должны быть сдвинуты, если v сдвигается – обозначим $L(v)$. Как уже было сказано, для каждого $3 \leq k \leq n$ обозначим как $v_1 = w_1, w_2, ..., w_t = v_2$ последовательность вершин $C_{k–1}$, когда проходим по часовой стрелке, и $w_p, ..., w_q$ как подпоследовательность вершин всех вершин из $C_{k–1}$, которые смежны с v_k. Назовём w_p *левым креплением* v_k и w_q – *правым*. Заметим, что $w_{p+1}, ..., w_{q–1}$ покрываются v_k. Для двух точек на сетке $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ мы обозначим как $\mu(P_1, P_2)$ пересечение двух линий с наклоном ± 1, проходящих одна через P_1, а другая через P_2, то есть:

$$\mu(P_1, P_2) = \left(\frac{x_2 + x_1 + y_2 - y_1}{2}, \frac{x_2 - x_1 + y_2 + y_1}{2}\right)$$

Изначально установим $P(v_1) = (–1, 0)$, $P(v_2) = (1, 0)$, $P(v_3) = (0, 1)$; также определим множества сдвига $L(v_i) = \{v_i\}$, $1 \leq i \leq 3$. Для каждого $4 \leq k \leq n$ мы допускаем, что планарное изображение с прямыми рёбрами $\Gamma_{k–1}$ графа $G_{k–1}$ было построено при соблюдении следующих условий:

1. $P(v_1) = (–((k – 1) – 2), 0)$ и $P(v_2) = ((k – 1) – 2, 0)$;
2. $x(w_1) < x(w_2) < ... < x(w_{t–1}) < x(w_t)$;
3. каждый сегмент $P(w_i)P(w_{i+1})$, $1 \leq i \leq t – 1$, имеет наклон или 1 или –1.

Теперь покажем, как добавить $P(v_k)$ к Γ_{k-1} и получить плоское изображение с прямыми рёбрами Γ_k для G_k.

1. Для каждой $v \in \bigcup_{i=1}^{p} L(w_i)$ ставим $x(v) = x(v) - 1$. Это шаг передвигает влево на 1 все вершины с внешней грани от w_1 до w_p и все вершины, находящиеся в зависимости от предыдущих.
2. Для каждого $v \in \bigcup_{i=q}^{t} L(w_i)$ ставим $x(v) = x(v) + 1$. Это шаг передвигает вправо на 1 все вершины с внешней грани от v_q до w_t и все вершины, находящиеся в зависимости от предыдущих.
3. Установим $P(v_k) = \mu(P(w_p), P(w_q))$. Этот шаг ставит вершину v_k так, что она может быть присоединена прямыми к своим соседям.
4. Установим $L(v_k) = \{v_k\} \cup (\bigcup_{i=p+1}^{q-1} L(w_i)$. Этот шаг определяет $L(v_k)$ как объединение v_k и множеств сдвига вершин, покрываемых v_k.

Практическая часть

Для работы с планарными графами была реализована программа в среде IDEA на языке Java, позволяющая проверять граф на планарность путём Гамма-алгоритма, генерировать случайный двусвязный планарный граф или строить плоское изображение планарного графа, разбивая его на двусвязные компоненты. В программе будет использоваться формат *graph6*. Graph6 – формат для компактного представления неориентированных графов. Пользовательский интерфейс в программе оформлен в виде кнопок полей, надписей и рисунков. Он просто и понятен, работа состоит в том, чтобы ввести входные данные и нажать соответствующую кнопку. При проверке планарности результат будет выведен в текстовое поле рядом с полем ввода, при генерации графов результаты будут выведены в поле в правом углу экрана, при визуализации графа для каждой компоненты двусвязности будет нарисована укладка в отдельном окне.

Программа может быть использована для учебных целей, для генерации планарных графов для других программ, а также просто для изучений свойств того или иного графа.

ЗАКЛЮЧЕНИЕ

Теория графов как наука находит все большее применение в современной жизни. С момента решения задачи кёнигсбергских мостов прошло много времени, и графы перестали уже быть чем-то несерьезным, стали привлекать все большее внимание специалистов по всему миру. Графам стали находить применение в картографии, в социологии, в химии, в схемотехнике и, само собой, в информатике и математике. Таким образом, теория графов является перспективной областью математики, требующей дальнейшего изучения.

В данной работе были рассмотрены основные понятия и определения из теории графов, были изучены планарные графы и теоремы, связанные с ними. Были представлены алгоритм проверки планарности Гамма-алгоритм, устанавливающий возможность быть уложенным без пересечений, алгоритм генерации случайного планарного графа на основе генераторов Больцмана и алгоритм Фрейззе – Паха – Поллака для визуализации планарного графа. В практической части работы представлена программа на языке Java в среде IDEA, позволяющая проверить граф на планарность и вывести итог, сгенерировать нужное число планарных графов с заданным числом вершин и вывести результаты генерации, а также визуализировать плоскую укладку планарного графа. Программа может быть использована для тестирования других программ, работающих с графами, для изучения свойств тех или иных графов, для их визуализации.

Планарность графа является довольно важной характеристикой, она принимается во внимание, например, при проектировании сетей или в информационной безопасности при создании систем с графическим паролем.