А. Н. Б а у ш е в (Санкт-Петербург, ПГУПС)**. Об алгоритмах построения паросочетаний в случайных графах.**

Пусть $G = \langle V, E\rangle$ — граф с множеством вершин V и множеством ребер E. Непустое подмножество ребер графа G, в котором любые два ребра не являются смежными, называется *независимым*. Максимальные относительно операции включения независимые подмножества ребер называются *паросочетаниями* в графе G. Паросочетания наибольшей мощности называются *наибольшими*, а их мощность — *числом паросочетания* графа G. Паросочетания, покрывающие все вершины графа G (имеющего четное число вершин), называются *полными* или *совершенными*.

Задача построения наибольших и полных сочетаний в графе G имеет многочисленные приложения и длинную богатую событиями историю [1]. В настоящее время известно несколько алгоритмов точного решения этих задач [1]. Все эти алгоритмы являются довольно сложными, и, несмотря на то, что для каждого из них имеется полиномиальная оценка зависимости времени работы алгоритма от порядка рассматриваемого графа, для практических задач они оказываются слишком медленными, если речь идет о больших сетях. По этой причине в последние три десятилетия широко разрабатываются эвристические алгоритмы решения этих задач [2, 3]. Эвристические алгоритмы быстро работают и во многих случаях дают точное решение задачи. Однако более или менее полное описание условий, налагаемых на рассматриваемый граф, при которых конкретный эвристический алгоритм находит точное решение, как правило, является сложной задачей.

Основным результатом в предлагаемом докладе является анализ работы алгоритма (A) построения паросочетания, одна итерация которого состоит из следующих трех шагов.

Шаг 1. Находим в графе G вершину $v \in V$ наименьшей положительной степени. Если таких вершин несколько, то выбираем любую из них, если же степени всех вершин равны нулю, то останавливаем алгоритм.

Шаг 2. Среди вершин, смежных с вершиной v, находим вершину u наименьшей степени. Если таких вершин несколько, то выбираем любую из них.

Шаг 3. Включаем ребро $[u, v]$ в строящееся паросочетание, удаляем вершины v и u из графа G и возвращаемся к шагу 1.

Время работы этого алгоритма допускает оценку $\tau(n, m) = O(m \log n)$ при $n \to \infty$, где n — число вершин графа G, а m — число ребер. Основное время при этом уходит на предварительную сортировку для каждой вершины списков смежных с ней вершин по степеням.

Можно построить пример последовательности графов G_n, $n = 1, 2, \ldots$, в которой номер графа равен его порядку, и такую, что разность между числом паросочетания графа G_n и мощностью паросочетания, построенного описанным выше алгоритмом, неограниченно возрастает с ростом n. Однако имеет место следующий результат.

Теорема. *Пусть G — граф четного порядка, имеющий гамильтонов цикл. Тогда результатом работы алгоритма A является полное паросочетание в графе G.*

Д о к а з а т е л ь с т в о. этой теоремы представляет собой логический анализ возможных ситуаций, возникающих после применения итерации алгоритма A. Наличие гамильтонова цикла длины n позволяет доказать, что шаги 2–3 будут осуществлены $n/2$ раз.

Естественными объектами для исследования эвристических алгоритмов являются случайные графы. Мы ограничимся рассмотрением случайных графов в классической модели Эрдеша–Реньи $G(n; p)$, в которой каждое из $N = n(n-1)/2$ возможных ребер независимо от остальных ребер присутствует в графе G с вероятностью p $(0 < p < 1)$.

Пусть $\omega(n)$ $(n = 1, 2, \ldots)$ — произвольная последовательность положительных чисел, сколь угодно медленно растущая к бесконечности; $p_\omega(n) = (\log n + \log\log n + \omega(n))/n$.

Следствие. *Пусть* $n = 2, 4, 6, \ldots$, *для всех* $p = p(n) \geqslant p_\omega(n)$ *достаточно больших* n, G — *случайный граф в модели* $G(n;p)$, $\mu_n = \mu_n(G)$ — *мощность паросочетания, построенного в графе* G *алгоритмом* A. *Тогда* $\mathbf{P}\{\mu_n = n/2\} = 1 + o(n)$ *при* $n \to \infty$.

Следствие непосредственно вытекает из хорошо известных [3] условий наличия гамильтоновых циклов у графов в модели $G(n;p)$ и приведенной теоремы.

В заключение рассмотрим кратко «жадный» алгоритм, в котором на каждой итерации произвольное ребро включается в строящееся паросочетание, а концевые вершины удаляются из графа. По-видимому, это самый быстрый из возможных алгоритмов построения паросочетаний. Простая структура жадного алгоритма делает сравнительно легким его анализ для графов в модели $G(n;p)$. В частности, нетрудно видеть, что для $\nu_n = \nu_n(G)$ — мощности паросочетания, построенного в графе G четного порядка жадным алгоритмом и $p \neq 1 + o(1)$ при $n \to \infty$, имеет место соотношение $\mathbf{P}\{\nu_n = n/2\} = o(1)$. Однако, несмотря на этот факт, вычислительные эксперименты показывают, что для достаточно плотных графов жадный алгоритм также строит полное или близкое к нему паросочетание. Например, для пятидесяти реализаций графов в модели $G(4000; 0,7)$ жадный алгоритм в 33 случаях построил полное паросочетание, а в остальных 17 случаях — паросочетание мощности 1999. Алгоритм A для этих 17 случаев построил полное паросочетание. Реализации графов имели около 5600000 ребер. При этом жадный алгоритм, реализованный в пакете MATLAB, работал в среднем около 1,33 секунды, в то время как алгоритм A — около 22 секунд (на компьютере со средними параметрами).

СПИСОК ЛИТЕРАТУРЫ

1. *Ловас Л.*, *Пламмер М.* Прикладные задачи теории графов. Теория паросочетаний в математике, физике, химии. М.: Мир, 1998.
2. *Avis D.* A survey of heuristics for the weighted matching problem. — Networks, 1983, v. 13, p. 475–493 [448].
3. *Bollobás B.* Random graphs. London: 2001.