Параллельные алгоритмы моделирования комплексных сетей*

С.В. Иванов, И.И. Колыхматов, А.В. Бухановский

В работе предлагается семейство параллельных алгоритмов, которые позволяют воспроизводить комплексную сеть как стохастический граф с заданными вероятностными свойствами. Каждый из алгоритмов основывается на естественных свойствах распараллеливания структуры графов, и соответствует особенностям конкретной вычислительной архитектуры – с общей памятью (SMP), кластерной или P2P. Приводятся результаты экспериментальных исследований производительности параллельных алгоритмов на многоядерных вычислительных системах и кластерной системе TForge-Mini. Работоспособность алгоритмов проиллюстрирована на примере моделирования эпидемиологических комплексных сетей, описывающих распространение ВИЧ.

1. Введение

Моделирование сетей на основе аппарата теории графов является важным направлением дискретной математики [1]. В последнее десятилетие возрос интерес исследователей к *комплексным сетям* с очень большим количеством узлов, структура которых нерегулярна, сложна и динамически эволюционирует во времени [2]. Примерами комплексных сетей служат социальные сети (знакомств, соавторства ученых [3]), информационные (цитирования в научных статьях [4], ссылок WWW [5]), технологические (Интернет как сеть компьютеров, транспортные и электрические сети) и биологические (сети нейронов в мозге, взаимодействующих протеинов, генетические сети). Для исследования таких комплексных сетей *in-silico*[1] требуется генерировать стохастические графы с порядком вершин 10^6–10^{10}. Это, в свою очередь, инициирует развитие параллельных алгоритмов генерации графов и соответствующих способов их хранения в распределенной вычислительной среде.

В настоящее время сформулировано четыре основных подхода к моделированию комплексных сетей: случайные пуассоновские графы и обобщенные случайные графы [6, 7]; марковские случайные графы и модель блуждания по «графу графов» с вероятностями, пропорциональными желаемым свойствам [8–10]; модель «малого мира» Уотса и Строгатса [11] и ее обобщения, эволюционная модель роста сети Барабаси и Альберт [12] и модель Прайса [4]. Первые три подхода подразумевают генерацию случайного графа с известным заранее числом вершин и заданными вероятностными свойствами. В данной работе предлагаются параллельные алгоритмы генерации комплексных сетей с заданным законом распределения степеней вершин, адаптированные для параллельных вычислительных архитектур с общей (SMP) и разделенной (MPP) памятью.

2. Вероятностная модель комплексной сети

Комплексная сеть представляется в виде графа G, который определяется как совокупность (V, E) конечного множества вершин V, $\dim(V) = N$, и множества ребер E, состоящего из неупорядоченных пар (u, v), где $u, v \in V$ и $u \neq v$. Каждая вершина характеризуется своей степенью, т.е. числом инцидентных ей ребер. Упорядоченный список степеней вершин называется степенной последовательностью.

* Работа выполнена при частичной финансовой поддержке в рамках Европейского проекта VIROLAB (EU FP6 # INFSO-IST-027446).

[1] В современной терминологии eScience – компьютерное моделирование объектов, другие способы исследования которых (наблюдением (in vivo, in situ), активным экспериментом (in vitro)) не представляются целесообразными или возможными.

Интегральной характеристикой комплексной сети является закон распределения степеней вершин p_k, который определяет вероятность того, что случайно выбранная вершина имеет степень k. Степенную последовательность для неориентированного графа удобно представить в форме $d = (c_1^{k_1}, c_2^{k_2}, ..., c_p^{k_p})$, где числа c_i суть степени, а показатель k_i означает количество повторений числа c_i в последовательности[1]. Такая запись позволяет связать дискретное распределение степеней вершин p_k со степенной последовательностью d в форме $p_k \overset{def}{=} P[x = c_i] = k_i / N$. В общем случае подразумевается, что степенная последовательность является монотонно невозрастающей, однако в случае генерации комплексных сетей данное требование не является обязательным.

В модели случайных графов [6] ребро, инцидентное любым двум вершинам, присутствует или отсутствует с равной вероятностью, вследствие чего распределение p_k будет биномиальным или (в пределе по N) пуассоновским. Однако большинство реальных сетей имеет структуру, отличную от структуры случайных графов, что отражается на характере распределения степеней вершин [2]. В частности, во многих реальных сетях эмпирическое распределение степеней вершин интерпретируется в терминах степенного распределения $p_k = k^{-\gamma} / \zeta(\gamma)$, где ζ-функция Римана играет роль нормирующей константы. Это распределение характеризуется единственным параметром γ, определяющим скорость убывания хвоста распределения. В ряде случаев могут использоваться более сложные аппроксимации на основе степенного распределения, например, на основе усеченного распределения или смеси нескольких распределений с разными параметрами.

3. Способ генерации комплексной сети по заданному закону распределения степеней вершин

Наиболее популярным способом генерации сетей с заданным законом распределения степеней вершин является т.н. *конфигурационная* модель [13–15] или близкая ей по смыслу l-процедура [16]. Если распределение p_k задано, то вычислительная процедура сводится к следующим операциям:

- формируется степенная последовательность d, выбирая N чисел k_i согласно заданному распределению p_k, где $i = \overline{1, N}$;
- каждой вершине i графа присваивается k_i «заготовок» (концов) для будущих ребер;
- из степенной последовательности случайно извлекаются пары «заготовок». Они соединяются ребром в том случае, если новое ребро не приведет к появлению ребер-циклов (петель) или мультиребер. Если ребро сгенерировано, то соответствующие индексы из степенной последовательности удаляются;
- предыдущий шаг повторяется до тех пор, пока степенная последовательность не пуста.

На основе распределения p_k любой граф может быть построен $\prod_i k_i!$ различными способами, поскольку «заготовки» для будущих ребер неразличимы. Таким образом, этот процесс с равной вероятностью генерирует любую возможную конфигурацию сети с заданным распределением степеней вершин. Преимуществом данного алгоритма является его универсальность, так как с его помощью можно построить сеть с любым распределением степеней вершин.

[1] Так, например $(3^1, 2^2, 1^5) = (3, 2, 2, 1, 1, 1, 1, 1)$.

4. Базовый параллельный алгоритм

Непосредственное распараллеливание конфигурационной модели не представляется целесообразным, поскольку проверка наличия мультиребер требует хранения для каждой вершины номеров всех смежных с ней вершин. Когда несколько вычислителей (процессоров или ядер) одновременно участвуют в создании новых ребер, для корректной работы алгоритма требуется обеспечить синхронизацию доступа вычислителей к разделяемым ресурсам, что с точки зрения параллельных вычислений является нежелательной операцией. В результате многочисленные синхронные обращения к спискам смежных вершин могут серьезно уменьшить эффективность параллельной реализации алгоритма. Разумной альтернативой является блочная организация алгоритмов, основанная на операциях с максимально большими блоками комплексной сети.

Поскольку комплексная сеть, задаваемая произвольным распределением p_k, является статистически однородной, то количество ребер между двумя блоками сети одинакового размера должно быть одинаковым; следовательно, для любой пары блоков ребра можно генерировать независимо. Если количество таких блоков превышает количество вычислителей, то, выбирая независимые пары блоков для каждого вычислителя, генерацию сети можно осуществлять параллельно. Для системы, состоящей из K вычислителей, наибольшие независимые блоки обеспечит деление множества вершин N на $2K$ равных частей. При этом между вершинами каждой пары различных блоков будет $M/2K^2$ ребер, а между вершинами одного блока – $M/4K^2$ ребер, где $M = N\sum_i k_i p_i = N\langle k \rangle$ общее число ребер в сети, и $\langle k \rangle$ – средняя степень вершины. На рис. 1 приведена общая схема параллельного алгоритма генерации сети.

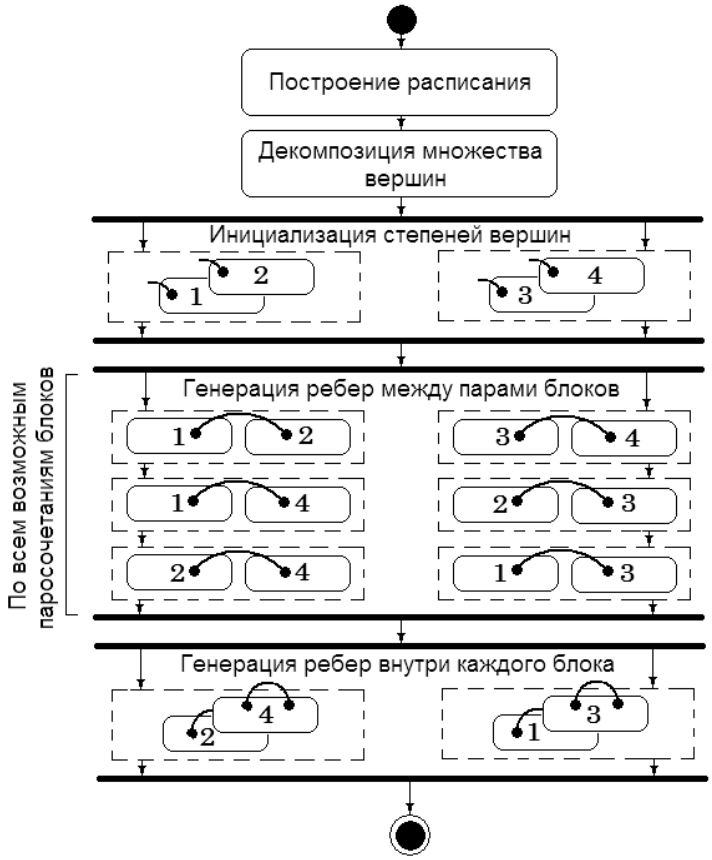

Рис. 1. Общая схема параллельного алгоритма генерации комплексной сети

Первый этап базового параллельного алгоритма генерации комплексной сети состоит в формировании независимых фрагментов степенной последовательности d_j, $j = \overline{1, 2K}$, и инициализации блоков сети.

На втором этапе генерируются связи между вершинами из разных блоков. Множество из $2K$ блоков разбивается на K пар так, чтобы связи между вершинами любых двух фиксированных блоков генерировались лишь однажды. При этом за каждым вычислителем закрепляются два независимых блока. В ходе фазы каждый процессор выполняет построение ребер графа по алгоритму конфигурационной модели, на каждом шаге выбирая случайную вершину из одного блока своей пары, а другую – из другого блока. Этот процесс повторяется $2K-1$ раз до тех пор, пока не будут установлены ребра между всеми возможными сочетаниями блоков.

Третий этап заключается в генерации ребер между вершинами каждого из $2K$ блоков. Связи между вершинами из одного блока генерируются независимо от других блоков, поэтому этот этап можно рассматривать как реализацию последовательного алгоритма конфигурационной модели разными вычислителями одновременно, применительно к независимым блокам (по два блока на вычислитель).

Принципиальной особенностью рассмотренного выше алгоритма является наличие расписания, которое обеспечивает обработку всех возможных парных сочетаний блоков. Существуют различные способы построения расписаний [17]; степень их применимости зависит, в первую очередь, от архитектуры вычислительной системы. В частности, для MPP-систем принципиальное значение имеет балансировка расписания таким образом, чтобы этап обменов данными занимал минимально возможное время. В данном случае допустимо использовать циклические расписания, аналогичные применяемым в матричных алгоритмах Фокса и Кэннона [18].

5. Отображение алгоритма на вычислительную архитектуру

5.1. Система с общей памятью (SMP)

Отображение алгоритма, изображенного на рис. 1, на архитектуру вычислительной системы с общей памятью не требует принудительного разделения данных между вычислителями. Алгоритм реализуется через набор потоков, каждый из которых на произвольной итерации работает с собственными блоками данных. По завершении каждой итерации выполняется синхронизация, после чего со всеми потокам ассоциируются новые блоки в соответствии с расписанием, см. рис. 2а. Программная реализация такого алгоритма выполнена на основе технологии OpenMP [19].

Учитывая, что время работы параллельного алгоритма, изображенного на рис. 2а, линейно зависит от N, параллельное ускорение в первом приближении может быть представлено в форме модификации закона Амдала:

$$S_K = \frac{1+g(\langle k\rangle)}{1+\dfrac{g(\langle k\rangle)}{K}}, \qquad (1)$$

где безразмерная функция $g(\langle k\rangle)=\alpha+\beta\langle k\rangle$ отражает зависимость времени выполнения от средней степени вершины $\langle k\rangle$. Коэффициент α характеризует ресурсоемкость операции установления нового ребра, а коэффициент β определяет затраты на просмотр списка вершин в поиске мультиребер. На рис. 3а приведены результаты вычислительных экспериментов на компьютере с четырехъядерным процессором Intel Core2 Quad 2.4 GHz в сопоставлении с выражением (1) при $\alpha=0{,}91$ и $\beta=0{,}054$ для различных значений $\langle k\rangle$. Из рис. видно, что модель (1), несмотря на простоту, удовлетворительно согласуется с экспериментальными данными. При этом значения параллельного ускорения весьма невелики: на четырех узлах не превышают 2. Это связано с достаточно весомой (по сравнению с операцией над ребрами) процедурой декомпозиции множества вершин и создания заготовок.

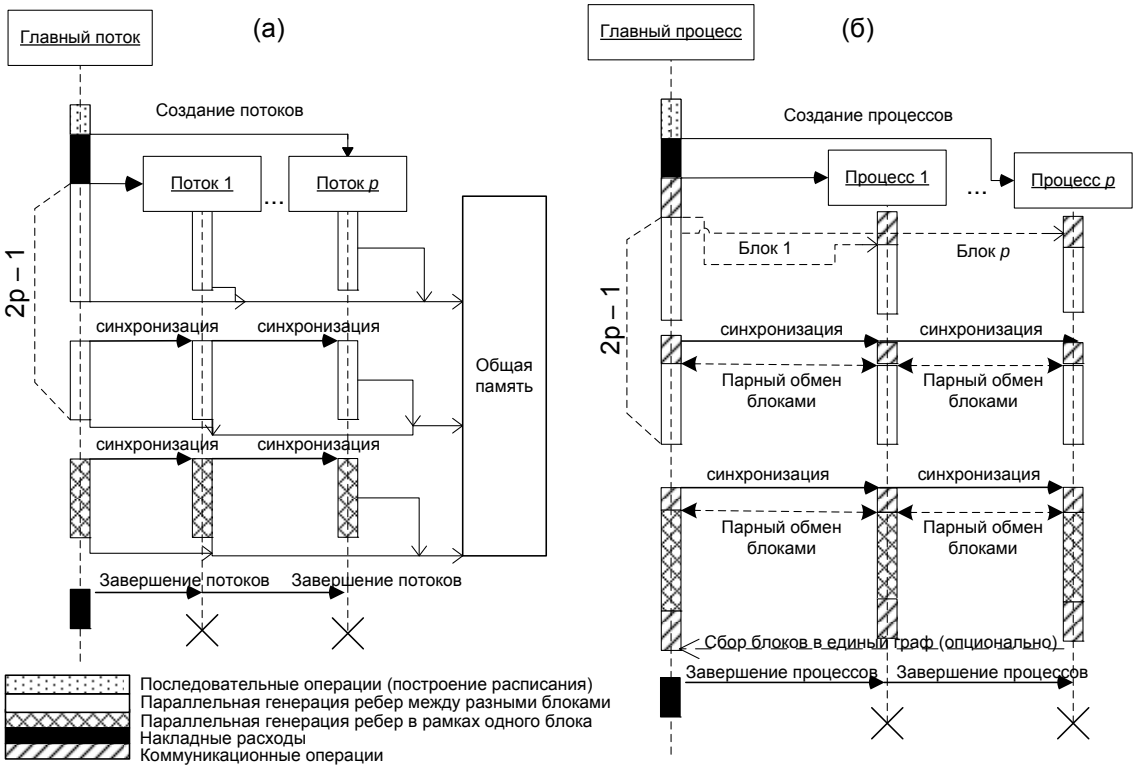

Рис. 2. Схема алгоритма в виде UML-диаграммы последовательностей: (а) для SMP-систем; (б) для MPP-систем

5.2. Система с разделенной памятью (MPP)

Для систем с разделенной памятью реализация параллельного алгоритма требует физического распределения фрагментов сети между вычислителями, с последующей организацией обменов между узлами в соответствии с заданным расписанием. Программная реализация выполнена с использованием технологии MPI, см. рис. 2б.

Главной проблемой при реализации предложенного параллельного алгоритма при помощи MPI является необходимость передачи между узлами списков смежности. Функции MPI предназначены для передачи данных, расположенных в памяти последовательно. Эффективная реализация списков смежности подразумевает независимый динамический массив для каждой вершины отдельно. По этой причине для передачи списков смежности разумно использовать упаковку и распаковку динамических данных при помощи специальных функций MPI_Pack/MPI_Unpack.

Технология MPI позволяет эффективно работать как с разделенной, так и с общей памятью; в последнем случае скорость обменов данными кардинально возрастает. Этот эффект необходимо учитывать в том случае, если программа функционирует на системе гибридной архитектуры, например, кластере на основе SMP-узлов. В качестве примера на рис. 3б приведены оценки параллельного ускорения при разных значениях $\langle k \rangle$ по результатам расчетов на кластере TForge-Mini (4 двухпроцессорных двухъядерных узла на основе AMD Opteron 275). С учетом накладных расходов на обмен данных между узлами, модель (1) трансформируется в более привычную для MPP-систем форму:

$$S_K = \frac{K}{1 + \dfrac{(K-1)}{1+g(\langle k \rangle)} + \chi(K-K_0)\dfrac{(K-K_0)(K-K_0-1)}{1+g(\langle k \rangle)}\tau}. \qquad (2)$$

Здесь K_0 – максимальное количество вычислителей (ядер) на узле, $\chi(\bullet)$ – ступенчатая функция Хевисайда, τ – безразмерный параметр, характеризующей отношение базовой операции по передаче данных к базовой вычислительной операции. Величина τ не зависит от других параметров задачи и (для заданного алгоритма) является объективной характеристикой вычислительной архитектуры. Для кластера TForge-Mini $\tau \cong 0{,}55$. На рис. 3б модель (2) сопоставлена с результатами экспериментов; видно, что результаты согласуются удовлетворительно. Сопоставляя рис. 3а и 3б, заметно, что при одних и тех же данных алгоритм для общей памяти масштабируется хуже, чем для разделенной памяти. Это объяснимо не только различием архитектур (и скорости доступа к общей памяти) на узле кластера и в рабочей станции с четырехъядерным процессором, но и различием программных реализаций. Для SMP-реализации (в силу отсутствия необходимости физической передачи данных между узлами) хранение списков смежности организовано более эффективно, чем для MMP-реализации. Как следствие абсолютное время последовательной работы MPP-реализации в среднем в 1,5 раза больше, чем для SMP-реализации, что способствует повышению зернистости задачи, и, как следствие, увеличению масштабируемости.

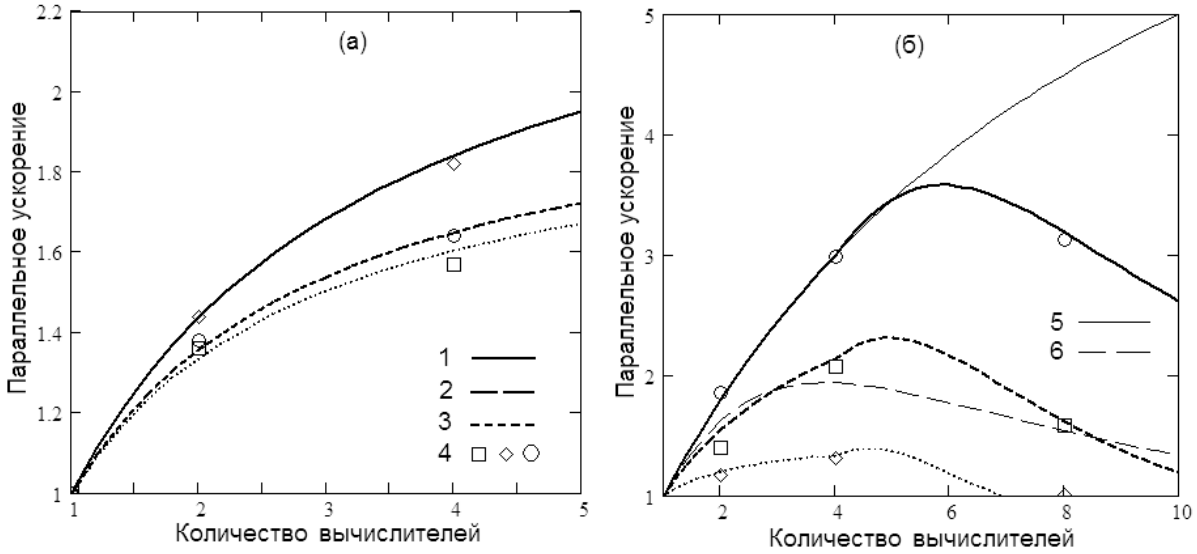

Рис. 3. Параллельное ускорение алгоритма генерации комплексной сети: (а) для SMP-систем; (б) для MPP-систем. 1- модель, $\langle k \rangle = 12$, 2- модель, $\langle k \rangle = 3{,}6$, 3 - модель, $\langle k \rangle = 1{,}8$, 4 – экспериментальные данные; 5 – модель (2) для полной SMP-системы ($K_0 = 16$), 6 – модель (2) для полной MPP-системы ($K_0 = 1$)

6. Приложения

В качестве иллюстрации на рис. 4 представлен пример визуализации стохастических графов, отображающих сети сексуальных контактов для гомо- (рис. 4а) и гетеросексуальной (рис. 4б) популяций. Эти сети смоделированы на основании статистических оценок распределения p_k по данным исследований [20]. Визуализация выполнена с использованием пакета Pajek; для укладки графа использован алгоритм Фрухтермана-Рейнгольда [21]. Из рис. видно, что в случае малых значений параметра распределения γ (рис. 4а) подавляющее большинство вершин графа образуют один большой связный кластер, в случае же больших значений γ (рис. 4б) помимо одного связного кластера в структуре графа присутствует большое число пар вершин, связанных лишь друг с другом. Рассмотренные алгоритмы и полученные на их основе результаты использованы авторами в целях моделирования динамики эпидемии ВИЧ [22].

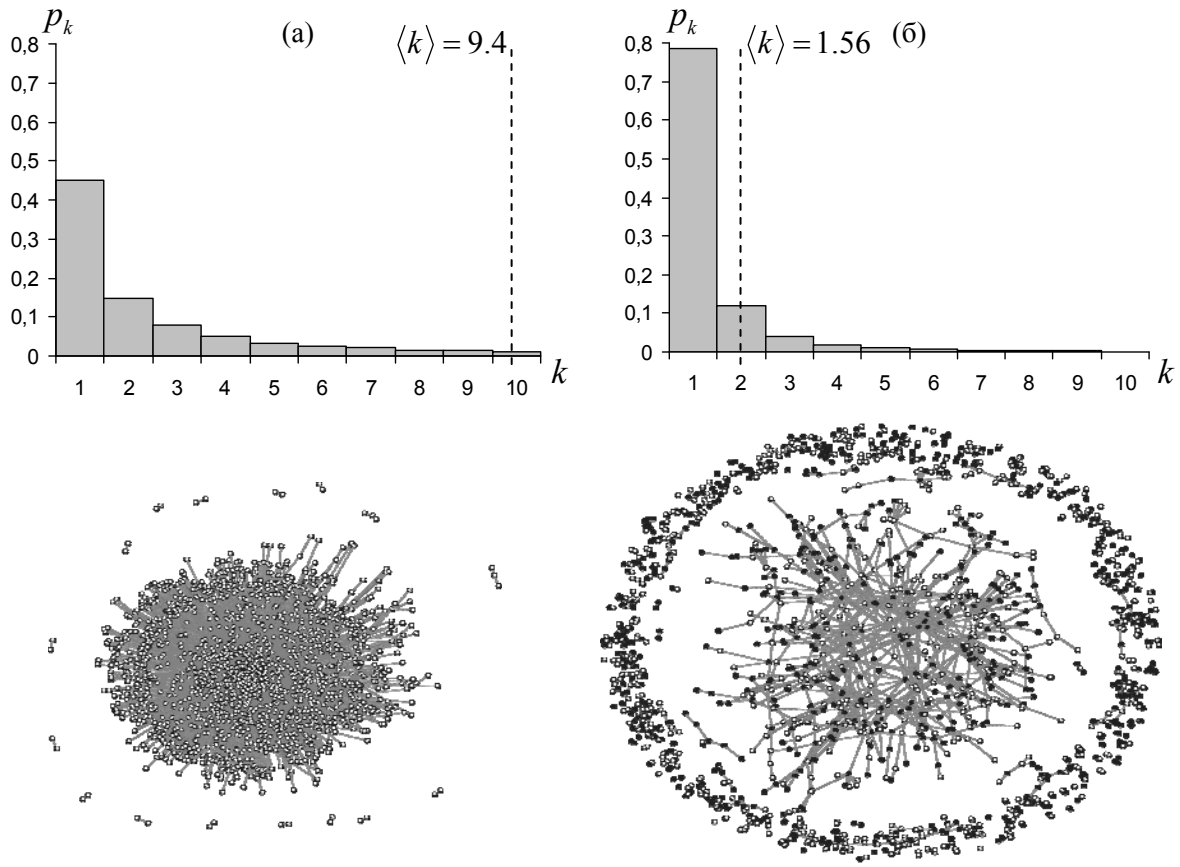

Рис. 4. Примеры графов, построенных для различных степенных последовательностей: (а) $\gamma = 1.6$; (б) $\gamma = 2.7$

7. Заключение

Таким образом, предложенный параллельный алгоритм генерации комплексной сети с заданным законом распределения допустимо использовать на вычислительных системах как с SMP-, так и с MPP-архитектурой. Возможности современных рабочих станций (пусть даже на основе многоядерных процессоров) ограничены построением сетей размера порядка 10^7 вершин. Потому моделирование более обширных сетей возможно лишь на основе кластерных архитектур. Несмотря на то, что эффективность данного алгоритма невелика (по результатам экспериментов – не более 50%), его принципиальным качеством является практическая независимость масштабируемости от размера комплексной сети N. Это дает определенные гарантии возможности получения результата за разумное (пусть и достаточно большое время) при увеличении количества вычислительных узлов.

Литература

1. Boccaletti S., Latora V., Moreno Y., Chavez M., Hwang D.-U. Complex networks: Structure and dynamics // Physics Reports. — 2006. — Vol. 424, N. 4–5. — P. 175–308.
2. Newman M.E.J. The Structure and Function of Complex Networks // SIAM Review. — 2003. — Vol. 45, N. 2. — P. 167–256.
3. Redner S. How popular is your paper? An empirical study of the citation distribution // European Physical Journal B. — 1998. — Vol. 4. — P. 131–134.

4. Price D.J. de S. A general theory of bibliometric and other cumulative advantage processes // Journal of the American Society for Information Science. — 1976. — Vol. 27. — P. 292–306.
5. Broder A., Kumar R., Maghoul F. et al. Graph structure in the web // Computer Networks. — 2000. — Vol. 33. — P. 309–320.
6. Erdős P., Rényi A. On the evolution of random graphs // Publications of the Mathematical Institute of the Hungarian Academy of Sciences. — 1960. — Vol. 5. — P. 17–61.
7. Rapoport A. Contribution to the theory of random and biased nets // Bulletin of Mathematical Biophysics. — 1957. — Vol. 19. — P. 257–277.
8. Holland P.W., Leinhardt S. An exponential family of probability distributions for directed graphs // Journal of the American Statistical Association. — 1981. — Vol. 76. — P. 33–65.
9. Frank O., Strauss D. Markov graphs // Journal of the American Statistical Association. — 1986. — Vol. 81. — P. 832–842.
10. Gkantsidis C., Mihail M., Zegura E. The markov chain simulation method for generating connected power law random graphs // Fifth Workshop on Algorithm Engineering and Experiments, January 11, 2003, Baltimore, Maryland, USA, Proceedings. SIAM. — 2003. — P. 16–25.
11. Watts D.J., Strogatz S.H. Collective dynamics of “small-world” networks // Nature. — 1998. — Vol. 393. — P. 440–442.
12. Barabási A.-L., Albert R. Emergence of scaling in random networks // Science. — 1999. — Vol. 286. — P. 509–512.
13. Bekessy A., Bekessy P., Komlos J. Asymptotic enumeration of regular matrices // Studia Scientiarum Mathematicarum Hungarica. — 1972. — Vol. 7. — P. 343–353.
14. Bender E.A., Canfield E.R. The asymptotic number of labeled graphs with given degree sequences // Journal of Combinatorial Theory A. — 1978. — Vol. 24. — P. 296–307.
15. Molloy M., Reed B. A critical point for random graphs with a given degree sequence // Random Structures Algorithms. — 1995. — Vol. 6. — P. 161–179.
16. Емеличев В.А., Мельников О.И., Сарванов В.И. и др. Лекции по теории графов. — М.: Наука, 1990. — 384 с.
17. Lee P. Efficient algorithms for data distribution on distributed memory parallel computers // IEEE Transactions on Parallel and Distributed Systems. — 1997. — Vol. 8, N. 8. — P. 825–839.
18. Choi J., Dongarra J.J., Walker D.W. Parallel matrix transpose algorithms on distributed memory concurrent computers // Parallel Computing. — 1995. — Vol. 21, N. 9. — P. 1387–1405.
19. Chandra R., Menon R., Dagum L. et al. Parallel Programming in OpenMP. — Morgan Kaufmann Publishers, 2000. — 231 p.
20. Schneeberger A., Mercer C.H., Gregson S.A. et al. Scale-free networks and sexually transmitted diseases: a description of observed patterns of sexual contacts in Britain and Zimbabwe // Sexually Transmitted Diseases. — 2004. — Vol. 31. — P. 380–387.
21. Batagelj V., Mrvar A. Pajek – analysis and visualization of large networks // Connections. — 1998. — Vol. 21. — P. 47–57.
22. Sloot P.M.A., Ivanov S.V., Boukhanovsky A.V. et al. Stochastic simulation of HIV population dynamics through complex network modeling // International Journal of Computer Mathematics. — In press, 2007.