

GRAPH PARTITIONING ALGORITHMS: LITERATURE REVIEW

A. R. Gerb*, G. A. Omarova**

*Novosibirsk State University,
630090, Novosibirsk, Russia

**Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2023-3-19-36

EDN: LXQVVD

This paper is devoted to the analysis of modern graph partitioning algorithms. We observe exact solutions, sequential, iterative, multilevel, streaming and parallel algorithms and note advantages and disadvantages of some algorithms.

Key words: graphs, coarsening, serial and parallel algorithms, optimization.

References

1. Itogi tretego kvartala 2022 goda sotcseti Vkontakte. [Electron. Res.]: <https://vk.com/press/q3-2022-results>.
2. Annual Report 2022 (SEC Filing Form 10-K). Meta Platforms, Inc. [Electron. Res.]: <https://d18rn0p25nwr6d.cloudfront.net/CIK-0001326801/e574646c-c642-42d9-9229-3892b13aabfb.pdf>.
3. Annual Report 2022 (SEC Filing Form 10-K). Twitter, Inc. [Electron. Res.]: <https://www.sec.gov/ix?doc=/Archives/edgar/data/1418091/000141809122000029/twtr-20211231.htm>.
4. Gottesbüren L. et al. Shared-memory n-level hypergraph partitioning // Proc. of the Symp. on algorithm engineering and experiments (ALENEX). Soc. for Industrial and Appl. Math., 2022.
5. Hamers L. et al. Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula // Inform. Process. and Manag. 1989. V. 25, iss. 3. P. 315–318.
6. Bourne F., Lelarge M., Vojnovic M. Balanced graph edge partition // Proc. of the 20th ACM SIGKDD Intern. conf. on knowledge discovery and data mining. 2014.
7. Brandes U. et al. On modularity clustering // IEEE Trans. On Knowledge And Data Engineering. 2007.
8. Geri M., Johnson D. Vychislitelnye mashiny i trudnoreshaemye zadachi. M.: Mir, 1982.
9. Armbruster M. Branch-and-cut for a semidefinite relaxation of large-scale minimum bisection problems. PhD-thesis. Technische Universität Chemnitz. Chemnitz, 2007.
10. Delling D. et al. Exact combinatorial branch-and-bound for graph bisection // Proc. of the 14th Workshop on Algorithm Engineering and Experiments (ALENEX). Soc. for Industrial and Applied Mathematics, 2012.

The work was supported by the Ministry of Science and Higher Education of the Russian Federation (project code 0251-2020-0001).

11. Felner A. Finding optimal solutions to the graph partitioning problem with heuristic search // Ann. Math. Artif. Intell. 2005. V. 45. P. 293–322.
12. Feldmann A. E., Widmayer P. An $O(n^4)$ time algorithm to compute the bisection width of solid grid graphs // Proc. of the Algorithms–ESA 2011: 19th Annual European Symp., Saarbrücken, Germany, Sept. 5–9, 2011. Springer Berlin Heidelberg, 2011.
13. Hager W. W., Phan D. T., Zhang H. An exact algorithm for graph partitioning. Mathematical Programming. 2013. V. 137. P. 531–556.
14. Karisch S. E., Rendl F., Clausen J. Solving graph bisection problems with semidefinite programming // INFORMS J. on Comput. 2000. V. 12, iss. 3. P. 177–191.
15. Sellmann M., Sensen N., Timajev L. Multicommodity flow approximation used for exact graph partitioning // Algorithms–ESA 2003: 11th Annual European Symp., Budapest, Hungary, Sept. 16–19, 2003. Proc. 11. Springer Berlin Heidelberg, 2003.
16. Fiduccia C. M., Mattheyses R. M. A linear-time heuristic for improving network partitions // Proc. of the 19th Conf. on Design Automation. 1982. P. 175–181.
17. Bui T. N., Jones C. A heuristic for reducing fill-in in sparse matrix factorization // Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (United States), 1993.
18. Walshaw C. An exploration of multilevel combinatorial optimization / Multilevel Optimization in VLSICAD. Combinatorial Optimization. V. 14. Springer, Boston, MA, 2003.
19. Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM Journal on scientific Computing. 1998. V. 20, N 1. P. 359–392.
20. Gerb A. R., Omarova G. A. Применение теории графов в алгебраических методах решения разрешенных SLAU // Пробл. информ. 2022. № 3. С. 77–89.
21. Akhremtsev Y., Sanders P., Schulz C. High-quality shared-memory graph partitioning // IEEE Trans. on Parallel and Distributed Systems. 2020.
22. Çatalyürek Ü. V., Aykanat C. Patoh (partitioning tool for hypergraphs). Encyclopedia of parallel computing. Springer, 2011.
23. Preen R. J., Smith J. Evolutionary n-level hypergraph partitioning with adaptive coarsening // IEEE Trans. on Evolutionary Computation. 2019. V. 23, iss. 6. P. 962–971.
24. Sanders P., Schulz C. Engineering multilevel graph partitioning algorithms // Algorithms–ESA 2011: 19th Annual European Symp., Saarbrücken, Germany, Sept. 5–9, 2011. Proc. 19. Springer Berlin Heidelberg, 2011.
25. Hamann M., Strasser B. Graph bisection with Pareto optimization // Journal of Experimental Algorithmics (JEA). 2018. V. 23. P. 1–34.
26. Henzinger A., Noe A., Schulz C. ILP-based local search for graph partitioning // ACM J. of Experimental Algorithmics (JEA). 2020. V. 25. P. 1–26.
27. Godenschwager C. et al. A framework for hybrid parallel flow simulations with a trillion cells in complex geometries // Proc. of the Intern. conf. on high performance computing, networking, storage and analysis. 2013.
28. Alpert C. J., Huang J. H., Kahng A. B. Multilevel circuit partitioning // Proc. of the 34th Annual design automation Conf. 1997.
29. Osipov V., Sanders P. n-Level graph partitioning // Algorithms–ESA 2010: 18th Annual European Symp., Liverpool, UK, Sept. 6–8, 2010. Proc., Part I 18. Springer Berlin Heidelberg, 2010.
30. Schlag S. et al. High-quality hypergraph partitioning // ACM J. of Experimental Algorithmics. 2023. V. 27. P. 1–39.
31. Alpert C. J. The ISPD98 circuit benchmark suite // Proc. of the 1998 Intern. symp. on physical design. 1998.
32. Garey M. R., Johnson D. S. Computers and intractability. San Francisco: Freeman, 1979.
33. Heuer T., Maas N., Schlag S. Multilevel Hypergraph Partitioning with Vertex Weights Revisited. arXiv preprint arXiv:2102.01378. 2021.

34. Patwary M. A. K., Garg S., Kang B. Window-based streaming graph partitioning algorithm // Proc. of the Australasian Computer Science Week Multiconf. 2019.
35. Faraj M. F., Schulz C. Buffered streaming graph partitioning. arXiv preprint arXiv:2102.09384. 2021.
36. Jafari N., Selvitopi O., Aykanat C. Fast shared-memory streaming multilevel graph partitioning // J. of Parallel and Distributed Computing. 2021. V. 147, iss. 2. P. 140–151.
37. Stanton I., Kliot G. Streaming graph partitioning for large distributed graphs // Proc. of the 18th ACM SIGKDD Intern. conf. on knowledge discovery and data mining. 2012.
38. Slota G. M. et al. Scalable, multi-constraint, complex-objective graph partitioning // IEEE Trans. on Parallel and Distributed Systems. 2020. V. 31, iss. 1. P. 2789–2801.
39. Zhang W., Chen Y., Dai D. AKIN: A streaming graph partitioning algorithm for distributed graph storage systems. 2018 18th IEEE/ACM Intern. Symp. on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2018.
40. Tsourakakis C. et al. Fennel: Streaming graph partitioning for massive scale graphs // Proc. of the 7th ACM Intern. conf. on Web search and data mining. 2014.
41. Çatalyürek Ü. V. et al. Multithreaded clustering for multi-level hypergraph partitioning // IEEE 26th Intern. Parallel and Distributed Proc. Symp. IEEE, 2012.
42. Meyerhenke H., Sanders P., Schulz C. Partitioning complex networks via size-constrained clustering // Experimental Algorithms: 13th Intern. Symp., SEA 2014, Copenhagen, Denmark, June 29 – July 1, 2014. Proc. 13. Springer International Publishing, 2014.
43. LaSalle D., Karypis G. Multi-threaded graph partitioning. 2013 IEEE 27th Intern. Symp. on Parallel and Distributed Processing. IEEE, 2013.
44. Maleki S. et al. Bipart: a parallel and deterministic hypergraph partitioner // Proc. of the 26th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming. 2021.
45. Slota G. M., Madduri K., Rajamanickam S. Complex network partitioning using label propagation // SIAM J. Scien. Comput. 2016. V. 38, iss. 5.
46. Gottesbüren L. et al. Scalable shared-memory hypergraph partitioning // Proc. of the Workshop on algorithm engineering and experiments (ALENEX). Soc. for Industrial and Appl. Math., 2021.
47. Gottesbüren L., Hamann M. Deterministic parallel hypergraph partitioning // Euro-Par 2022: Parallel Processing: 28th Intern. conf. on parallel and distributed computing, Glasgow, UK, Aug. 22–26, 2022. Proc. Cham: Springer International Publishing, 2022.
48. Savage J. E., Wloka M. G. Parallelism in graph-partitioning // Journal of Parallel and Distributed Computing. 1991. V. 13, iss. 3. P. 257–272.
49. Gottesbüren L. et al. Deep multilevel graph partitioning. arXiv preprint arXiv:2105.02022. 2021.
50. Hamann M. et al. Distributed graph clustering using modularity and map equation // Euro-Par 2018: Parallel Processing: 24th Intern. Conf. on Parallel and Distributed Computing, Turin, Italy, Aug. 27–31, 2018. Proc. Springer International Publishing, 2018.
51. Andersen R., Peres Y. Finding sparse cuts locally using evolving sets // Proc. of the 41st Annual ACM symp. on theory of computing. 2009.
52. Back T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press, 1996.
53. Barnard S. T., Simon H. D. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurrency: Practice and experience. 1994.
54. Benlic U., Hao J. K. An effective multilevel memetic algorithm for balanced graph partitioning // 22nd IEEE Intern. conf. on tools with artificial intelligence. IEEE. 2010.
55. Boppana R. B. Eigenvalues and graph bisection: An average-case analysis // 28th Annual Symp. on Foundations of Computer Science (sfcs 1987). IEEE, 1987.

56. Donath W. E., Hoffman A. J. Lower bounds for the partitioning of graphs // IBM J. of Research and Development. 1973.
57. Fiedler M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory // Czechoslovak Math. J. 1975. V. 25, iss. 4. P. 619–633.
58. Grigni M., Manne F. On the complexity of the generalized block distribution. LNCS. 1996. V. 1117.
59. LaSalle D. et al. Improving graph partitioning for modern graphs and architectures // Proc. of the 5th Workshop on Irregular Applications: Architectures and Algorithms. 2015.
60. Manne F., Søorevik T. Partitioning an array onto a mesh of processors // Applied parallel computing industrial computation and optimization: 3rd Intern. Workshop, PARA'96 Lyngby, Denmark, Aug. 18–21, 1996. Proceedings 3. Springer Berlin Heidelberg, 1996.
61. Meyerhenke H., Sanders P., Schulz C. Partitioning (hierarchically clustered) complex networks via size-constrained graph clustering // J. of Heuristics. 2016. V. 22, iss. 1. P. 759–782.
62. Sanders P., Schulz C. Distributed evolutionary graph partitioning // Proc. of the 14th workshop on algorithm engineering and experiments (ALENEX). Soc. for Industrial and Appl. Math., 2012.
63. Timothy A. Davis and Yifan Hu. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38, 1, Article 1 (December 2011), 25 pages. DOI: 10.1145/2049662.2049663, 2011
64. Ugander J., Backstrom L. Balanced label propagation for partitioning massive graphs // Proc. of the sixth ACM Intern. conf. on Web search and data mining. 2013.
65. Van Bevern R. et al. On the parameterized complexity of computing balanced partitions in graphs // Theory of Computing Systems. 2015. V. 57, iss. 1. P. 1–35.
66. Yaşar A. et al. On symmetric rectilinear matrix partitioning. arXiv preprint arXiv:2009.07735. 2020.
67. Yaşar A., Çatalyürek Ü. V. Heuristics for symmetric rectilinear matrix partitioning. arXiv preprint arXiv:1909.12209. 2019.



АЛГОРИТМЫ РАЗБИЕНИЯ ГРАФОВ: ОБЗОР ЛИТЕРАТУРЫ

А. Р. Герб*, Г. А. Омарова**

*Новосибирский государственный университет,
630090, Новосибирск, Россия

**Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

УДК 519.17+519.61

DOI: 10.24412/2073-0667-2023-3-19-36

EDN: LXQVVD

Работа посвящена разбору современных методов и алгоритмов разбиения графов. Исследованы и проанализированы точные решения, последовательные итерационные, многоуровневые, потоковые и параллельные алгоритмы. Отмечены как преимущества, так и слабые места алгоритмов, выявленные при их реализации.

Ключевые слова: графы, огрубление, последовательные и параллельные алгоритмы, многоуровневые алгоритмы.

Введение. Для решения многих современных задач при обработке данных используются графы. Наиболее ярким примером обработки больших данных могут служить социальные сети. Так, ежемесячная аудитория социальной сети «ВКонтакте» в 3-м квартале 2022 г. составляла 76,9 млн активных пользователей [1]; на декабрь 2022 г. аудитория Facebook* составила 2,96 млрд пользователей [2]; ежедневная аудитория Twitter* в январе 2023 г. составляла 368 млн активных пользователей с более чем 1,47 млрд социальных связей [3]. Помимо пользователей, в социальных сетях существуют дополнительные объекты, которые взаимодействуют с пользовательскими узлами: сообщества, страницы событий и т. д.

Примеры больших наборов данных графов не ограничиваются социальными сетями: многие сложные биологические системы могут быть смоделированы с помощью графов. Примерами могут служить межбелковые взаимодействия и сети работы с молекулами ДНК. В этих сетях узлами являются биологические сущности (такие как гены и белки), а ребра соответствуют их общему участию в каком-либо биологическом процессе. Такие процессы могут варьироваться от простых прямых взаимодействий до более сложных отношений, в которых участвуют более двух сущностей.

Несмотря на большие размеры графов, необходимо выполнять вычисления с данными, например: вычислять PageRank (алгоритм ссылочного ранжирования), транслировать обновления Twitter, определять ассоциации белков, а также выполнять множество других задач. Большие графы могут состоять из терабайтов сжатых данных при хранении на

Работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации (код проекта 0251-2020-0001)

*запрещенная в РФ организация — прим. ред.

дисках, что является слишком большим объемом данных для одной машины. Чтобы эффективно обрабатывать такие графы, общепринятым решением является распределение данных по большому количеству машин и использование параллельных, распределенных алгоритмов. Такой подход порождает множество проблем системной инженерии, из которых мы рассмотрим только проблему распределения данных, которая в случае графового представления сводится к проблеме разбиения вершин графа на подмножества. Цель состоит в том, чтобы минимизировать количество пересекающихся ребер из разных подмножеств, сохраняя при этом примерно одинаковое количество вершин (или ребер) в каждом подмножестве.

Графы, которые встречаются на практике, во многих случаях не являются случайными. Ребра демонстрируют большую локальность (географическая близость вершин в социальных сетях, связь по теме или домену в Интернете и т. п.). Эта локальность позволяет предположить, что в реальных графах существует хорошее разбиение или, по крайней мере, разбиения, которые значительно лучше случайных разрезов. Коммуникация между разными вычислительными машинами, даже внутри одной локальной сети, значительно дороже межпроцессорной – задержка в сети измеряется в микросекундах, в то время как межпроцессорная связь измеряется в наносекундах. Такое несоответствие существенно замедляет обработку, когда необходимо использовать сеть. Для больших графов объем перемещаемых данных может достигать гигабайтов, что приводит к перегрузке сетевых каналов.

Основная проблема при разбиении сложных данных графа заключается в том, что сложно создать линейное упорядочение данных, которое сохраняет локальность ребер, такое упорядочение может вообще не существовать. Один из первых методов разбиения графа на два блока используется и сегодня, это спектральная бисекция. Спектральные методы были впервые использованы У. Донатом и А. Хоффманом [4], и М. Фидлером [5], а затем усовершенствованы другими учеными [6, 7]. Спектральная бисекция позволяет получить глобальную информацию о связности графа путем вычисления собственного вектора, соответствующего второму наименьшему собственному значению лапласиана графа L .

Однако спектральные методы не подходят для больших данных. Во-первых, это связано с большими вычислительными затратами. Во-вторых, существующие формулы требуют полной информации о графе. Когда граф физически не помещается в памяти одной ЭВМ, поддержание последовательного представления всего состояния невозможно. Это привело к появлению локальных методов спектрального разбиения, но локальные методы все еще требуют доступа к большим частям графа, полагаются на сложную распределенную координацию и большие вычисления после загрузки данных. Таким образом, мы ищем новый тип решения.

Пусть $G = (V, E)$ – граф, построенный на основе матрицы A размерности $n \times n$, где V – множество всех вершин, $|V| = n$, E – множество взвешенных ребер графа. Вес каждого ребра e_{ij} совпадает с коэффициентом, стоящим в матрице A на i -й строке и в j -м столбце. Основная цель задачи состоит в том, чтобы найти близкое к оптимальному сбалансированное разбиение множества V на непересекающие подмножества с минимально возможными вычислениями.

1. Основные методы и алгоритмы. Проблема разбиения графа является NP -полней [8]. Решения обычно находятся с помощью эвристики и приближенных алгоритмов, и за время существования проблемы было разработано множество алгоритмов, которые

позволяют найти достаточно хорошее разбиение. Например, существуют потоковые алгоритмы, предназначенные для разрезания больших графов, с одной стороны, они выполняются быстро и потребляют мало памяти, но при этом дают решения низкого качества по сравнению с другими алгоритмами. С другой стороны, существует широкий спектр последовательных алгоритмов, которые решают задачи разбиения графов без использования параллелизма, но достаточно часто имеют параллельные аналоги. Существуют также эволюционные алгоритмы, которые затрачивают много ресурсов для достижения еще более высокого качества. Время выполнения таких алгоритмов очень большое, поэтому обычно они используются только на графах с несколькими сотнями тысяч узлов. Распределенные параллельные алгоритмы хорошо масштабируются для больших графов, но если они не реализуют многоуровневые стратегии, их качество намного ниже, чем у последовательных алгоритмов разбиения. Тем не менее, преимущества этого подхода заключаются в том, что огромные графы могут быть разбиты очень быстро, он также позволяет разбивать графы на машинах, не обладающих большой вычислительной мощностью.

1.1. Точные решения. Существует большое количество работ, посвященных методам оптимального решения задачи разбиения графа. Сюда входят методы, посвященные случаю двудольного разбиения [9–15], и методы, решающие общие задачи. Большинство методов опирается на метод ветвей и границ. Границы получаются с помощью различных подходов: С. Кариш [14] и М. Армбрусттер [9] используют полуопределенное программирование (Semidefinite programming (SDP)). Другим подходом для решения проблемы воспользовался У. Хагер в своих работах [13], где сформулирована задача разбиения в виде непрерывной квадратичной задачи, к которой применяется метод ветвей и границ.

В целом, в зависимости от используемого метода, можно наблюдать две альтернативы: либо полученные границы очень хороши и дают небольшие деревья с ветвями и связями, но их трудно вычислить; либо границы несколько слабее и дают большие деревья, но вычисляются быстрее. Последнее имеет место при использовании комбинаторных ограничений. На связных подграфах двумерной решетки проблема двудольного разбиения может быть решена с трудоемкостью $O(n^4)$ [12]. Все вышеописанные методы, как правило, могут решать только очень небольшие графы при очень большом времени работы, при этом если они и могут решать большие графы двудольных разбиений с умеренными затратами времени [10], то сильно зависят от ширины бисекций графа. Более того, экспериментальная оценка этих методов рассматривает только малые числа подмножеств $k \leq 4$.

1.2. Последовательные алгоритмы. Алгоритм Кернигана – Лина (Kernighan – Lin, KN) является итерационным. Для работы он требует наличие изначального разбиения графа. На каждой итерации он ищет подмножества вершин графа, такие что обмен вершин между ними приводит к уменьшению разреза. Алгоритм работает до тех пор, пока удается отыскать такие подмножества. Каждая итерация алгоритма KL имеет трудоемкость $O(|E| \log |E|)$ [6]. Также существует модификация данного алгоритма – алгоритм Фидуччии и Маттейса [16], который уменьшает трудоемкость до $O(|E|)$. Алгоритм FM использует две очереди приоритетов, по одной на долю. Первоначально добавляют каждую граничную вершину в очередь приоритетов вместе с выигрышем от перемещения вершины в другую долю. На каждом шаге повторяется перемещение с наибольшим выигрышем, которое не нарушает ограничение баланса, а затем обновляется выигрыш всех соседей, не подвергшихся перемещению. Итерация заканчивается, когда все вершины перемещены или ограничение баланса не допускает дальнейших перемещений. Алгоритм FM также выполняет перемещения с отрицательным коэффициентом усиления и поэтому способен

выходить из локального оптимума. В конце концов он возвращается к наилучшему решению во время прохода.

Алгоритм KL находит локально оптимальные разделы, когда начинает с хорошего начального разбиения и когда средняя степень графа велика [17]. Если хорошее начальное разбиение неизвестно, алгоритм KL повторяется для разных случайно выбранных начальных разбиений, и выбирается то, которое дает наименьший разрез по ребрам. Многократное повторение может быть дорогостоящим, особенно если граф большой. Однако, поскольку разбивается только самый меньший грубый граф [17], выполнение нескольких прогонов требует очень мало времени.

1.3. Многоуровневая парадигма. Наиболее успешным подходом к решению проблемы разбиения графа является многоуровневая парадигма. Она состоит из трех этапов: **огрубление, начальное разбиение, последовательное уточнение разбиения**. На этапе огрубления создается последовательность уменьшающихся и структурно похожих графов. Как только граф становится достаточно маленьким, алгоритм начального разбиения получает разбиение самого грубого графа. На этапе улучшения разбиение проецируется на следующий более крупный граф в последовательности, и на каждом этапе алгоритмы локального поиска минимизируют количество разрезаемых ребер.

Огрубление. Этап огрубления направлен на вычисление последовательности более грубых приближений входного графа таким образом, чтобы сохранить его структурные свойства и действовать в некотором смысле как фильтр, который удаляет как можно больше ненужной информации из пространства поиска [18].

Возможно множество подходов к реализации стадии огрубления графа, например сжатие нескольких вершин в одну, в которой суммируется вес всех сжимаемых вершин. Для дальнейшего уменьшения размера более грубого графа можно также удалить все ребра, которые становятся идентичными, кроме одного, при котором их вес суммируется. Таким образом, можно получить разбиение с теми же свойствами баланса и размером разреза при проектировании разбиения на следующий более крупный граф в последовательности.

В упомянутых выше способах применяются паросочетания для поиска пар вершин или ребер для объединения. Однако в случае неравномерного распределения степеней существующие алгоритмы кластеризации могут уменьшать размер сложных графов более эффективно, чем подходы, основанные на паросочетаниях. В данном случае возникает проблема, связанная с тем, что вершины, инцидентные вершинам с наибольшей степенью, часто остаются несопоставленными. В работе [19] предложено несколько методов уменьшения количества несовпадающих вершин, например объединение двух несмежных вершин, если у них есть общий сосед или они смежны с двумя вершинами, которые уже сопоставлены.

В последние годы исследования были сосредоточены на методах кластеризации сложных сетей и улучшении процесса огрубления с помощью информации о структуре подмножеств графа. Некоторые такие подходы были рассмотрены в [20].

Начальное разбиение. Огрубление обычно продолжается до тех пор, пока не останется заранее выбранное число вершин. Часто используют многоуровневую рекурсивную бисекцию для получения начального разбиения самого грубого графа. Чтобы получить начальное двудольное разбиение, часто запускают набор различных алгоритмов несколько раз с последующим распространением меток или локальным поиском FM, начиная стадию улучшения с лучшим двудольным разбиением из этих запусков [19, 21, 22]. Р. Прин и Дж. Смит в работе [23] показали, что решение о прекращении огрубления часто зависит

от конкретного случая, и предложили адаптивный критерий остановки вместо использования одинакового постоянного числа вершин для всех случаев.

Улучшение разбиения. На данном этапе полученное разбиение самого грубого графа проецируется на все большие графы в последовательности, полученной на первом этапе. Каждая вершина более грубого графа в последовательности содержит отдельное подмножество вершин текущего графа, поэтому все это подмножество вершин будет принадлежать тому же подмножеству, что и вершина, содержащая их в более грубом графе.

Даже если текущее разбиение является локально минимальным разбиением текущего графа в цепочке, проецируемое разбиение на следующий граф может не быть локально минимальным. Поскольку предшествующий граф содержит больше вершин, после проектирования разбиения используются алгоритмы уточнения разбиения. Основная цель данных алгоритмов – выбрать два подмножества вершин, по одному из каждой части, такие, чтобы при обмене вершин из этих подмножеств уменьшить разрез.

В работе [24] П. Сандерс и К. Шульц предлагают метод уточнения на основе потока для двудольных разбиений. Идея заключается в выборе подмножества R вершин вокруг разреза бисекции $V = \{V_1, V_2\}$. Сеть потоков строится путем соединения $V_1 \setminus R$ с источником и $V_2 \setminus R$ со стоком. Полученный максимальный поток обеспечивает улучшенный разрез, который может нарушить ограничение баланса. Поэтому $|R|$ выбирается адаптивно, а вычисление сильно связных компонент используется для характеристики множества всех минимальных разрезов, которые затем можно искать для сбалансированного двудольного разбиения. Используя алгоритм FlowCutter [25], авторы более эффективно исследуют пространство решений.

В работе [26] используют целочисленное линейное программирование (ILP) для уточнения k подмножеств напрямую. Аналогично уточнению на основе потока, выбирается небольшая область вершин, которым разрешено перемещаться, остальные вершины сокращаются до одной супервершины на блок. Затем задача ILP формулируется на этом графе модели. Для ускорения ILP возможны предоставление начального эвристического решения и нарушение симметрии путем закрепления достаточно тяжелых вершин за соответствующими блоками.

Глубокое многоуровневое разделение. Разбиение на k подмножеств может быть получено как прямым разбиением на k подмножеств, так и применением рекурсивной бисекции. Рекурсивная бисекция была обобщена к многоуровневому разбиению [27]. В данном случае граф огрубляется до тех пор, пока не останется $2X$ вершин, где X – заранее заданное число, после чего выполняется бисекция самого грубого графа. На стадии улучшения подмножества, мощность которых превышает $2X$, будут повторно разбиваться на два, и так будет повторяться, пока число подмножеств не достигнет k .

n -уровневое графовое разбиение. Глубина многоуровневой иерархии обеспечивает компромисс между временем работы и качеством решения многоуровневого алгоритма [28]. Большее количество уровней огрубления предоставляет больше возможностей для уточнения текущего решения, но требует быстрых алгоритмов уточнения решения для приемлемого времени работы. Наиболее экстремальная версия многоуровневой парадигмы: сокращение одной вершины на каждом уровне, была изучена В. Осиповым в [29], а затем усовершенствована в работе [30] для разбиения гиперграфов.

Разбиение с неравномерно распределенными весами вершин. Многоуровневые алгоритмы включают методы предотвращения образования «тяжелых» вершин: ограничение максимально допустимого веса вершины, добавление веса вершины в огрубляющую

функцию в качестве штрафного члена. Если «тяжелые» вершины уже присутствуют на входе, то работа этих алгоритмов значительно затрудняется, как это происходит для многих реальных графов [31].

Существуют два подхода к вычислению сбалансированных разбиений при жестком ограничении баланса в многоуровневом разбиении: 1) алгоритм начального разбиения сразу ищет сбалансированное разбиение, и на стадии улучшения вершины перемещаются, только если они удовлетворяют ограничениям баланса; 2) в начальном разбиении и на некоторых итерациях стадии улучшения допускаются промежуточные нарушения баланса, при этом на стадии улучшения применяются методы перебалансировки для обеспечения сбалансированности конечного разбиения.

Учитывая, что нахождение сбалансированного разбиения на k подмножеств является NP -трудной задачей [32], оба подхода не гарантируют баланса. В работе [33] предложен подход, который позволяет алгоритмам, основанным на рекурсивном двудольном разбиении, надежно вычислять сбалансированные разбиения на практике. Идея заключается в том, чтобы предварительно назначить небольшую часть самых «тяжелых» вершин одному из двух блоков, рассматриваемых как фиксированные вершины, и оптимизировать разбиение на оставшихся вершинах.

1.4. Потоковые алгоритмы. Под потоковыми алгоритмами разбиения графов (Streaming Graph Partitioning, SGP) понимаются алгоритмы графа, удовлетворяющие следующим условиям: 1) получают на вход вершины или ребра графа в некотором последовательном порядке; 2) не могут хранить весь график в памяти; 3) выполняют постоянные решения по разбиению «на лету», основываясь на частичной информации о графике.

Алгоритмы потокового разбиения обычно следуют итеративной последовательности операций *load – compute – store*. Эта последовательность операций может использоваться для разбиения как потока вершин, так и потока ребер. Более того, она работает с пакетами, которые могут содержать как одну вершину/ребро, так и несколько одновременно.

Наиболее распространенной потоковой моделью является «однопроходная» модель, в которой вершины загружаются по одной вместе со списками смежности, а затем применяется некоторый алгоритм постоянного распределения их по множествам. Логика алгоритма может быть разной, это может быть функция хэширования или назначение вершины в блок с наивысшей оценкой на основе некоторой цели. Существуют и другие потоковые модели, например модель скользящего окна [34], в которой окно имеет фиксированный размер и содержит вершины, которые хранятся в памяти, чтобы помочь в принятии решений при проходе; или модель буферизованного потока [35, 36], в которой партии вершин последовательно загружаются и сразу разбиваются, а затем приписываются подмножествам и больше не перемещаются между ними.

Потоковые модели данных — одна из самых популярных тенденций в обработке больших данных. Алгоритмы SGP очень быстродейственные, они даже быстродейственнее, чем многоуровневые, однако дают решение более низкого качества. Тем не менее, многие приложения, требующие чрезвычайно быстрых методов переразбиения, все еще могут извлечь большую пользу из алгоритмов SGP, когда в качестве начального упорядочения предоставляется исходное решение, полученное более сильным алгоритмом.

И. Стэнтон и Г. Клиот в [37] предлагают однопроходный метод линейной детерминированной жадности (Linear Deterministic Greedy (LDG)), который производит решения с наилучшим общим разрезом. В данном алгоритме при назначении вершин приоритет отдается множествам, содержащим больше соседей, и используется штрафной множитель

для борьбы с дисбалансом; LDG разбивает граф с трудоемкостью $O(m + nk)$. Более того, авторы также предлагают простой однопроходный метод, основанный на хэшировании, который имеет трудоемкость $O(n)$ и производит плохое разрезание графа. Позднее Дж. Слота [38] рассмотрел проблему потокового вершинного разбиения с более теоретической точки зрения и доказал, что ни один алгоритм не может получить трудоемкость $O(n)$ при случайному или неблагоприятном упорядочении потока.

В работе [39] авторы предложили AKIN – потоковый алгоритм разбиения вершин для систем хранения распределенных графов. AKIN способен разбивать графы, в которых количество вершин заранее неизвестно, разрешая миграцию вершин между блоками с течением времени. Решения о назначении основаны на сходстве между вершинами, которое оценивается с помощью коэффициента Жаккара (Jaccard similarity coefficient) [5].

В работе [34] авторы предлагают алгоритм WStream – алгоритм потокового разделения вершин, который хранит в памяти скользящее окно. Авторы допускают несколько сотен вершин в скользящем окне, чтобы получить больше информации о вершине, прежде чем она будет назначена блоку на основе жадной функции. Как только вершина назначается множеству, из входного потока в скользящее окно загружается еще одна вершина, что позволяет поддерживать размер окна постоянным.

Авторы работы [36] предлагают использовать буферизованную потоковую модель. Предложенный алгоритм многократно загружает пакет вершин из потока, разбивает его, после чего распределяет вершины без их дальнейшего перераспределения. При работе используется однопроходный алгоритм LDG для огрубления, вычисления начального разбиения и его уточнения. Авторы распараллеливают LDG, разделяя вершины между процессорами, что позволяет распараллелить три этапа их многоуровневой схемы.

М. Фарадж и К. Шульц в работе [35] предлагают алгоритм HeiStream, который также разделяет вершины в буферизованной потоковой модели. Их алгоритм загружает партию вершин, строит модель графа, а затем разбивает эту модель с помощью многоуровневого алгоритма. В их графовой модели вершины из предыдущих партий, назначенные каждому блоку, представлены в виде одной большой вершины, закрепленной за соответствующим блоком. Для получения начального разбиения используется один проход алгоритма Fennel [40], а уточнение распространения меток также оптимизирует объективную функцию, используемую Fennel для распределения вершин по блокам.

1.5. Параллельные алгоритмы. В большей части первых работ по разбиению графов предполагалось использование последовательных алгоритмов. Эти алгоритмы были модифицированы для работы с распределенной памятью. В данном случае приложение уже имеет распределение графа, так как для масштабируемости памяти весь граф не хранится у каждого процессора. Таким образом, алгоритмы разбиения с распределенной памятью, как правило, не имеют глобального представления всего графа; они зачастую принимают решения о разбиении на основе частичных представлений локальных данных графа. В результате качество таких решений может оказаться ниже, чем у их последовательных аналогов, но, несмотря на это, алгоритмы с распределенной памятью очень важны для больших графов и для динамически адаптирующихся к изменяющимся нагрузкам приложений.

Большинство алгоритмов огрубления основано на жадном подборе или жадной кластеризации. Эти алгоритмы посещают вершины в некотором порядке, вычисляют выигрыш от присоединения к соседним группам или сравнения с соседом, а затем присоединяются к группе с самым высоким выигрышем. Параллельное посещение вершин дает хорошую

параллелизацию последовательных подходов. Проблема заключается в необходимости избегать несогласованных решений о кластеризации между потоками.

Также существуют параллельные многоуровневые алгоритмы. Отличительной особенностью данного типа алгоритмов является то, что они в основном пытаются максимально применять свою основную идею: 1) последовательное огрубление графа и разделение самого мелкого с последующим отображением разбиения на начальный граф; 2) оптимизировать под параллельное выполнение каждой из своих фаз в отдельности.

Ряд исследователей на протяжении нескольких десятилетий предпочитал использовать детерминированные параллельные алгоритмы ввиду простоты их отладки, производительности и воспроизводимости. Недостатком является меньшая гибкость в плане выбора дизайна алгоритмов и потенциальные накладные расходы на синхронизацию решений по оптимизации.

Фаза огрубления. В работе [41] авторы предлагают для агломеративной кластеризации и жадного подбора параллельные схемы, использующие блокировку, а также версию жадного подбора без блокировки, которая разрешает конфликты во время второго прохода. Алгоритмы, основанные на блокировке, сначала пытаются заблокировать посещенную вершину, вычисляют рейтинги, а затем перебирают кандидатов. Когда найден лучший кандидат, проверяется его блокировка, и, если он не заблокирован, старый кандидат заменяется, с него снимается блокировка. Для схемы с разрешением конфликтов без блокировки пары хранятся в глобальном массиве M без защиты на запись. После одного прохода вершины u , у которых $M[M[u]] \neq u$ (пара не совпала), сопоставляются сами с собой. Это те вершины, сопоставление которых было выполнено параллельным посещением и сопоставлением с другой вершиной.

В работе [21] используется параллельная кластеризация распространения меток с ограничением размера кластера [42] для более эффективного огрубления асимметричных входных данных. При этом не используются блокировки, поэтому возможно циклическое объединение кластеров. Ограничение на размер кластера гарантирует, что начальное разбиение сможет найти приемлемое разбиение. Размеры кластеров обновляются с помощью атомарных операций. Ограничение на размер проверяется перед обновлением, но это не обеспечивает атомарной согласованности. Поэтому результаты выполнения инструкций проверяются, чтобы гарантировать ограничение размера, и отменяются в случае его превышения. Для огрубления нет необходимости строго придерживаться ограничения на размер, в то время как для уточнения необходимо гарантировать баланс.

Фаза разбиения. Для начальных разбиений Д. Ласаль и Дж. Карипис [43] используют рекурсивную бисекцию. Каждый поток последовательно вычисляет бисекцию, из которых выбирается лучший вариант. Для рекурсии потоки статически разделяются на две группы, работающие над двумя отдельными подграфами. Позже, в [44], эта схема была усовершенствована, и вместо независимых последовательных расчетов потоки стали работать над одним разбиением, если граф достаточно большой.

В работе [45] использован параллельный поиск в ширину из k источников для вычисления k -мерного разбиения. Вершины присоединяются к блокам своих «родителей». В данном подходе не используется огрубление, что оправдывает использование параллельного алгоритма на данном этапе.

Фаза улучшения разбиения. При параллельном уточнении разбиения необходимо сохранять сбалансированность разбиения и гарантировать, что одновременные перемещения не приведут к его ухудшению. Методы, способные избегать локальных минимумов,

труднее распараллеливать, поскольку при их работе возникают промежуточные перемещения, которые на текущем шаге могут иметь отрицательный выигрыш, но такое перемещение может привести к лучшему разрезу в будущем.

Наиболее простым алгоритмом распараллеливания является распространение меток – простое параллельное посещение вершин. Хотя выигрыши могут быть неверными из-за одновременных перемещений в их окрестностях, на практике это не слишком сильно влияет на оптимизацию разреза. Для соблюдения баланса можно использовать атомарные инструкции для обновления весов блоков [21, 46]. Также для соблюдения баланса можно производить изменения в два этапа: на первом этапе собрать все потенциальные ходы, а на втором провести проверку и подтвердить перемещения [47, 44]. Во время уточнения важно обеспечить ограничение на размер путем проверки результата атомарной инструкцией, чтобы гарантировать сбалансированное разбиение.

Локализованный алгоритм FM [24] – это вариант FM, который начинается с нескольких граничных вершин и расширяет пространство поиска до соседей перемещенных вершин. Он хорош тем, что позволяет избегать локальных минимумов благодаря тому, что допускает перемещения с отрицательным выигрышем, но не тратит слишком много времени на бесперспективные области за счет коротких поисковых отрезков. Этот подход может быть распараллелен путем выполнения нескольких независимых локализованных поисков, в отличие от стандартного FM, который трудно эффективно распараллелить [48]. В работе [21] предлагается организовывать граничные вершины в очередь, которая случайным образом перемешивается. Потоки многократно опрашивают исходные вершины из очереди и выполняют локализованную FM вокруг своих исходных вершин. Перемещения не сообщаются другим потокам.

Перебалансировка. Если вершины перемещаются параллельно, то недостаточно проверить, сохранил ли баланс каждое отдельное перемещение, необходим механизм синхронизации. Поэтому некоторые алгоритмы уточнения не могут гарантировать сбалансированность разбиений, из-за чего возникает необходимость в явных алгоритмах перебалансировки. Более того, даже если некоторые алгоритмы уточнения могут гарантировать баланс, промежуточные нарушения баланса порой приводят к уменьшению разрезов после шага перебалансировки.

Дж. Карипис и соавторы включили ребалансировку в свою параллельную жадную доработку [43]. Локальные буферы перемещений потоков просматриваются в обратном порядке, отменяя перемещения в перегруженные блоки, при этом каждый поток восстанавливает избыточный вес, который был перемещен им в этот блок.

Для больших k в работе [49] использована одна очередь приоритетов на перегруженный блок с соотношением наибольшего выигрыша по отношению к неперегруженному блоку и веса вершины в качестве ключа. Каждая очередь заполняется достаточным количеством вершин для удаления избыточного веса в случае, когда целевые блоки становятся близкими к перегруженным. Параллелизм достигается путем параллельного опустошения различных перегруженных блоков. В работе [46] авторы допускают контролируемые нарушения баланса на этапе пересчета усиления локализованной FM в Mt-KaHuPar.

Авторы работы [4] предлагают n -уровневую версию Mt-KaHuPar. При последовательном алгоритме только одна вершина сокращается за один раз, но это обеспечивает хорошее качество решения благодаря высоколокализованному уточнению. В параллельной версии вершины организованы в виде «леса», и пары вершин сжимаются асинхронно. Лес дает условия предшествования (снизу вверх) для асинхронных сжатий. Благодаря этому вер-

шины в разных поддеревьях могут быть сжаты независимо друг от друга параллельно, как только закончатся их дочерние вершины.

Развертка (uncoarsening) вводит параллелизм путем параллельного разворачивания партий вершин $b > 1$. Партии строятся путем обхода леса в порядке сверху вниз, собирая сжатия, которые могут быть отменены независимо друг от друга в партии. Развертывание партии разрешает последние зависимости, необходимые для развертывания следующей партии. Вершины текущей партии служат исходными для высоколокализованного параллельного уточнения (распространение меток и FM).

Авторы [47] представляют модифицированную версию Mt-KaHuPar, использующую синхронные локальные перемещения [50]. Решения о перемещении не зависят от других перемещений в текущем раунде. Раунды далее разбиваются на подраунды, чтобы включить более свежую информацию. После каждого подраунда некоторые ходы одобряются, а некоторые отклоняются, например, из-за ограничения баланса или ограничения размера кластера для огрубления. Для уточнения используется обмен префиксами с наибольшим выигрышем между парами блоков. Чтобы учесть веса вершин, не являющихся единицами, вычисляются суммарные веса всех префиксов последовательностей перемещений, затем выбирается лучшая комбинация префиксов.

Список литературы

1. Итоги третьего квартала 2022 года соцсети Вконтакте. [Electron. Res.]: <https://vk.com/press/q3-2022-results>.
2. Annual Report 2022 (SEC Filing Form 10-K). Meta Platforms, Inc. [Electron. Res.]: <https://d18rn0p25nwr6d.cloudfront.net/CIK-0001326801/e574646c-c642-42d9-9229-3892b13aabfb.pdf>.
3. Annual Report 2022 (SEC Filing Form 10-K). Twitter, Inc. [Electron. Res.]: <https://www.sec.gov/ix?doc=/Archives/edgar/data/1418091/000141809122000029/twtr-20211231.htm>.
4. Gottesbüren L. et al. Shared-memory n-level hypergraph partitioning // Proc. of the Symp. on algorithm engineering and experiments (ALENEX). Soc. for Industrial and Appl. Math., 2022.
5. Hamers L. et al. Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula // Inform. Process. and Manag. 1989. V. 25, iss. 3. P. 315–318.
6. Bourne F., Lelarge M., Vojnovic M. Balanced graph edge partition // Proc. of the 20th ACM SIGKDD Intern. conf. on knowledge discovery and data mining. 2014.
7. Brandes U. et al. On modularity clustering // IEEE Trans. On Knowledge And Data Engineering. 2007.
8. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
9. Armbruster M. Branch-and-cut for a semidefinite relaxation of large-scale minimum bisection problems. PhD-thesis. Technische Universität Chemnitz. Chemnitz, 2007.
10. Delling D. et al. Exact combinatorial branch-and-bound for graph bisection // Proc. of the 14th Workshop on Algorithm Engineering and Experiments (ALENEX). Soc. for Industrial and Applied Mathematics, 2012.
11. Felner A. Finding optimal solutions to the graph partitioning problem with heuristic search // Ann. Math. Artif. Intell. 2005. V. 45. P. 293–322.
12. Feldmann A. E., Widmayer P. An $O(n^4)$ time algorithm to compute the bisection width of solid grid graphs // Proc. of the Algorithms–ESA 2011: 19th Annual European Symp., Saarbrücken, Germany, Sept. 5–9, 2011. Springer Berlin Heidelberg, 2011.
13. Hager W. W., Phan D. T., Zhang H. An exact algorithm for graph partitioning. Mathematical Programming. 2013. V. 137. P. 531–556.

14. Karisch S. E., Rendl F., Clausen J. Solving graph bisection problems with semidefinite programming // INFORMS J. on Comput. 2000. V. 12, iss. 3. P. 177–191.
15. Sellmann M., Sensen N., Timajev L. Multicommodity flow approximation used for exact graph partitioning // Algorithms-ESA 2003: 11th Annual European Symp., Budapest, Hungary, Sept. 16–19, 2003. Proc. 11. Springer Berlin Heidelberg, 2003.
16. Fiduccia C. M., Mattheyses R. M. A linear-time heuristic for improving network partitions // Proc. of the 19th Conf. on Design Automation. 1982. P. 175–181.
17. Bui T. N., Jones C. A heuristic for reducing fill-in in sparse matrix factorization // Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (United States), 1993.
18. Walshaw C. An exploration of multilevel combinatorial optimization / Multilevel Optimization in VLSICAD. Combinatorial Optimization. V. 14. Springer, Boston, MA, 2003.
19. Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM Journal on scientific Computing. 1998. V. 20, N 1. P. 359–392.
20. Герб А. Р., Омарова Г. А. Применение теории графов в алгебраических многосеточных методах для решения разреженных СЛАУ // Пробл. информ. 2022. № 3. С. 77–89.
21. Akhremtsev Y., Sanders P., Schulz C. High-quality shared-memory graph partitioning // IEEE Trans. on Parallel and Distributed Systems. 2020.
22. Çatalyürek Ü. V., Aykanat C. Patoh (partitioning tool for hypergraphs). Encyclopedia of parallel computing. Springer, 2011.
23. Preen R. J., Smith J. Evolutionary n-level hypergraph partitioning with adaptive coarsening // IEEE Trans. on Evolutionary Computation. 2019. V. 23, iss. 6. P. 962–971.
24. Sanders P., Schulz C. Engineering multilevel graph partitioning algorithms // Algorithms-ESA 2011: 19th Annual European Symp., Saarbrücken, Germany, Sept. 5–9, 2011. Proc. 19. Springer Berlin Heidelberg, 2011.
25. Hamann M., Strasser B. Graph bisection with Pareto optimization // Journal of Experimental Algorithmics (JEA). 2018. V. 23. P. 1–34.
26. Henzinger A., Noe A., Schulz C. ILP-based local search for graph partitioning // ACM J. of Experimental Algorithmics (JEA). 2020. V. 25. P. 1–26.
27. Godenschwager C. et al. A framework for hybrid parallel flow simulations with a trillion cells in complex geometries // Proc. of the Intern. conf. on high performance computing, networking, storage and analysis. 2013.
28. Alpert C. J., Huang J. H., Kahng A. B. Multilevel circuit partitioning // Proc. of the 34th Annual design automation Conf. 1997.
29. Osipov V., Sanders P. n-Level graph partitioning // Algorithms-ESA 2010: 18th Annual European Symp., Liverpool, UK, Sept. 6–8, 2010. Proc., Part I 18. Springer Berlin Heidelberg, 2010.
30. Schlag S. et al. High-quality hypergraph partitioning // ACM J. of Experimental Algorithmics. 2023. V. 27. P. 1–39.
31. Alpert C. J. The ISPD98 circuit benchmark suite // Proc. of the 1998 Intern. symp. on physical design. 1998.
32. Garey M. R., Johnson D. S. Computers and intractability. San Francisco: Freeman, 1979.
33. Heuer T., Maas N., Schlag S. Multilevel Hypergraph Partitioning with Vertex Weights Revisited. arXiv preprint arXiv:2102.01378. 2021.
34. Patwary M. A. K., Garg S., Kang B. Window-based streaming graph partitioning algorithm // Proc. of the Australasian Computer Science Week Multiconf. 2019.
35. Faraj M. F., Schulz C. Buffered streaming graph partitioning. arXiv preprint arXiv:2102.09384. 2021.
36. Jafari N., Selvitopi O., Aykanat C. Fast shared-memory streaming multilevel graph partitioning // J. of Parallel and Distributed Computing. 2021. V. 147, iss. 2. P. 140–151.

37. Stanton I., Kliot G. Streaming graph partitioning for large distributed graphs // Proc. of the 18th ACM SIGKDD Intern. conf. on knowledge discovery and data mining. 2012.
38. Slota G. M. et al. Scalable, multi-constraint, complex-objective graph partitioning // IEEE Trans. on Parallel and Distributed Systems. 2020. V. 31, iss. 1. P. 2789–2801.
39. Zhang W., Chen Y., Dai D. AKIN: A streaming graph partitioning algorithm for distributed graph storage systems. 2018 18th IEEE/ACM Intern. Symp. on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2018.
40. Tsourakakis C. et al. Fennel: Streaming graph partitioning for massive scale graphs // Proc. of the 7th ACM Intern. conf. on Web search and data mining. 2014.
41. Çatalyürek Ü. V. et al. Multithreaded clustering for multi-level hypergraph partitioning // IEEE 26th Intern. Parallel and Distributed Proc. Symp. IEEE, 2012.
42. Meyerhenke H., Sanders P., Schulz C. Partitioning complex networks via size-constrained clustering // Experimental Algorithms: 13th Intern. Symp., SEA 2014, Copenhagen, Denmark, June 29 – July 1, 2014. Proc. 13. Springer International Publishing, 2014.
43. LaSalle D., Karypis G. Multi-threaded graph partitioning. 2013 IEEE 27th Intern. Symp. on Parallel and Distributed Processing. IEEE, 2013.
44. Maleki S. et al. Bipart: a parallel and deterministic hypergraph partitioner // Proc. of the 26th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming. 2021.
45. Slota G. M., Madduri K., Rajamanickam S. Complex network partitioning using label propagation // SIAM J. Scien. Comput. 2016. V. 38, iss. 5.
46. Gottesbüren L. et al. Scalable shared-memory hypergraph partitioning // Proc. of the Workshop on algorithm engineering and experiments (ALENEX). Soc. for Industrial and Appl. Math., 2021.
47. Gottesbüren L., Hamann M. Deterministic parallel hypergraph partitioning // Euro-Par 2022: Parallel Processing: 28th Intern. conf. on parallel and distributed computing, Glasgow, UK, Aug. 22–26, 2022. Proc. Cham: Springer International Publishing, 2022.
48. Savage J. E., Wloka M. G. Parallelism in graph-partitioning // Journal of Parallel and Distributed Computing. 1991. V. 13, iss. 3. P. 257–272.
49. Gottesbüren L. et al. Deep multilevel graph partitioning. arXiv preprint arXiv:2105.02022. 2021.
50. Hamann M. et al. Distributed graph clustering using modularity and map equation // Euro-Par 2018: Parallel Processing: 24th Intern. Conf. on Parallel and Distributed Computing, Turin, Italy, Aug. 27–31, 2018. Proc. Springer International Publishing, 2018.
51. Andersen R., Peres Y. Finding sparse cuts locally using evolving sets // Proc. of the 41st Annual ACM symp. on theory of computing. 2009.
52. Back T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press, 1996.
53. Barnard S. T., Simon H. D. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurrency: Practice and experience. 1994.
54. Benlic U., Hao J. K. An effective multilevel memetic algorithm for balanced graph partitioning // 22nd IEEE Intern. conf. on tools with artificial intelligence. IEEE. 2010.
55. Boppana R. B. Eigenvalues and graph bisection: An average-case analysis // 28th Annual Symp. on Foundations of Computer Science (sfcs 1987). IEEE, 1987.
56. Donath W. E., Hoffman A. J. Lower bounds for the partitioning of graphs // IBM J. of Research and Development. 1973.
57. Fiedler M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory // Czechoslovak Math. J. 1975. V. 25, iss. 4. P. 619–633.
58. Grigni M., Manne F. On the complexity of the generalized block distribution. LNCS. 1996. V. 1117.

59. LaSalle D. et al. Improving graph partitioning for modern graphs and architectures // Proc. of the 5th Workshop on Irregular Applications: Architectures and Algorithms. 2015.
60. Manne F., Søorevik T. Partitioning an array onto a mesh of processors // Applied parallel computing industrial computation and optimization: 3rd Intern. Workshop, PARA'96 Lyngby, Denmark, Aug. 18–21, 1996. Proceedings 3. Springer Berlin Heidelberg, 1996.
61. Meyerhenke H., Sanders P., Schulz C. Partitioning (hierarchically clustered) complex networks via size-constrained graph clustering // J. of Heuristics. 2016. V. 22, iss. 1. P. 759–782.
62. Sanders P., Schulz C. Distributed evolutionary graph partitioning // Proc. of the 14th workshop on algorithm engineering and experiments (ALENEX). Soc. for Industrial and Appl. Math., 2012.
63. Timothy A. Davis and Yifan Hu. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38, 1, Article 1 (December 2011), 25 pages. DOI: 10.1145/2049662.2049663, 2011
64. Ugander J., Backstrom L. Balanced label propagation for partitioning massive graphs // Proc. of the sixth ACM Intern. conf. on Web search and data mining. 2013.
65. Van Bevern R. et al. On the parameterized complexity of computing balanced partitions in graphs // Theory of Computing Systems. 2015. V. 57, iss. 1. P. 1–35.
66. Yaşar A. et al. On symmetric rectilinear matrix partitioning. arXiv preprint arXiv:2009.07735. 2020.
67. Yaşar A., Çatalyürek Ü. V. Heuristics for symmetric rectilinear matrix partitioning. arXiv preprint arXiv:1909.12209. 2019.



Герб Артем Родионович — студент Новосибирского государственного университета. E-mail: gerb-artem@mail.ru; В 2021 году окончил механико-математический факультет НГУ, степень бакалавра. В настоящее время магистрант Новосибирского государственного университета.

Gerb Artem – NSU student. In 2021, he graduated from the faculty of mechanics and mathematics of Novosibirsk State University, bachelor's degree. Currently, he is a master's student at Novosibirsk State University.



Омарова Гульзирида Алимовна — канд. физ.-мат. наук, науч. сотр. Института вычислительной математики и математической геофизики СО РАН; e-mail: gulzira@rav.ssc.ru.

Гульзирида Омарова окончила механико-математический факультет Новосибирского государственного университета в 1995 году. В 1997 г. окончила магистратуру механико-математического факультета Новосибирского государственного университета. В 2007 году за-

щтила кандидатскую диссертацию по специальности 05.13.18 в Институте вычислительной математики и математической геофизики СО РАН. С 2000 года по настоящее время работает в Институте вычислительной математики и математической геофизики СО РАН. Г. А. Омарова занимается математическим моделированием в области сетей и графов и их применением в различных прикладных областях. Ею опубликовано более 40 работ в таких областях как построение и исследование алгоритмических моделей на сетях и графах, математическое и имитационное моделирование в исследовании различных прикладных задач.

Gulzira Omarova graduated from the faculty of mechanics and mathematics of Novosibirsk State University in 1995, received her M. S. degree in Mathematics from the State University of Novosibirsk (1997), Ph.D. degree in Computer Science (2007) from the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of the Russian Academy of Sciences (SB RAS). Since 2000 till present works at the Institute of Computational Mathematics and Mathematical Geophysics of SB RAS. G. Omarova is engaged in mathematical modeling in the field of networks and graphs, and their application in various application areas.

She has published more than 40 papers in such areas as the construction and study of algorithmic models on networks and graphs, mathematical and simulation modeling in the study of various applied problems.

Дата поступления — 08.06.2023