LD-Leiden: локальное параллельное обнаружение сообществ в динамических графах

Г. В. Боков*, А. Ю. Коновалов‡, А. С. Упорова§, С. В. Моисеев¶, И. А. Сафонов‖, А. А. Радионов**

Представлен параллельный алгоритм LD-Leiden обнаружения сообществ в динамических взвешенных графах. Он локально обновляет иерархию Leiden и согласует перемещения вершин. На девяти реальных графах (до 50,9 млн вершин и 117 млн ребер) геометрическое среднее ускорение относительно повторного запуска Leiden равно 25,64 при среднем отношении итоговых модулярностей 0,996. На графах до 214 млн вершин и 3,30 млрд ребер переход от 1 к 64 потокам дает среднее ускорение в 5,22 раза.

Ключевые слова: динамический граф, обнаружение сообществ, модулярность, Leiden.

Постановка задачи. Задача обнаружения сообществ заключается в разбиении вершин графа на группы так, чтобы плотность связей внутри групп была выше, чем между ними. В динамическом графе, где ребра добавляются, удаляются и меняют вес, разбиение необходимо уточнять после каждого пакета изменений. Поскольку изменение малой доли ребер может затронуть многие сообщества, требуется обновлять разбиение с меньшими затратами, чем при полном пересчете сообществ на всем графе, без существенной потери качества.

Пусть $G_t = (V_t, E_t, w_t)$ — граф после t-го пакета изменений, где V_t, E_t — множества вершин и ребер, $w_t : V_t \times V_t \to \mathbb{R}_{\geq 0}$ — веса ребер. Пакет

* *Боков Г. В.* — МГУ имени М.В.Ломоносова, e-mail: bokovgrigoriy@gmail.com.
Bokov Grigoriy — Lomonosov Moscow State University.
‡ *Коновалов А. Ю.* — МГУ имени М.В.Ломоносова, e-mail: alexandr.konoval@gmail.com.
Konovalov Aleksandr — Lomonosov Moscow State University.
§ *Упорова А. С.* — МГУ имени М.В.Ломоносова, e-mail: annuporova2003@gmail.com.
Uporova Anna — Lomonosov Moscow State University.
¶ *Моисеев С. В.* — МГУ имени М.В.Ломоносова, e-mail: stanislav.moiseev@gmail.com.
Moiseev Stanislav — Lomonosov Moscow State University.
‖ *Сафонов И. А.* — НИУ ВШЭ, e-mail: isafonov27@gmail.com.
Safonov Ivan — HSE University.
** *Радионов А. А.* — Московская лаборатория инфокоммуникационных технологий, e-mail: alex.radionov89@gmail.com.
Radionov Alexander — Moscow Infocommunication Technology Laboratory.

Опубликовано на условиях лицензии CC-BY 4.0
Published under CC-BY 4.0

обновлений $\Delta_t = \{(i, j, \delta_{ij}) : \delta_{ij} \neq 0\}$ задает суммарные приращения весов $|\Delta_t|$ пар вершин. Приращения δ_{ij} могут быть отрицательными, но уменьшение веса ребра $w_t(i, j) = w_{t-1}(i, j) + \delta_{ij}$ допускается лишь до нуля, что означает его удаление из графа.

Качество разбиения оценивается модулярностью [1], которую алгоритм стремится увеличить. Пусть $\mathcal{P}$ — разбиение V_t на сообщества, $c_i \in \mathcal{P}$ — сообщество вершины i. Модулярность задается формулой:

$$Q(\mathcal{P}) = \frac{1}{m} \sum_{i,j} \left(w_{ij} - \gamma \frac{k_i^{\text{out}} k_j^{\text{in}}}{m} \right) \mathbf{1}\{c_i = c_j\}, \tag{1}$$

где $w_{ij} = w_t(i, j)$, $m = \sum_{i,j} w_{ij} > 0$, $k_i^{\text{out}} = \sum_j w_{ij}$, $k_j^{\text{in}} = \sum_i w_{ij}$ — исходящие и входящие взвешенные степени, $\gamma > 0$ — параметр резолюции. При $\gamma = 1$ модулярность показывает нормированное на m превышение веса ребер внутри сообществ над ожидаемым в случайной модели с ожидаемыми весами $k_i^{\text{out}} k_j^{\text{in}}/m$ и ожидаемыми степенями, равными исходным.

Алгоритм LD-Leiden и его свойства. LD-Leiden между пакетами сохраняет и локально обновляет все уровни иерархии Leiden [2], тогда как GVE-Leiden [3] строит ее заново, а DF-Leiden [4] локализует поиск на исходном графе.

В LD-Leiden поиск начинается с концов измененных ребер, затронутых сообществ и соответствующих вершин следующих уровней. Для них и их соседей параллельно выбираются лучшие перемещения, увеличивающие модулярность. Этап согласования отбирает набор, в котором сообщество только отдает либо только принимает вершины. После пробного применения вычисляется точный прирост модулярности набора $\Delta Q = Q_{\text{after}} - Q_{\text{before}}$ по обновленным весам и степеням сообществ. При $\Delta Q \leq \delta\alpha Q_0$ набор отменяется, при $\Delta Q < \alpha Q_0$ перемещения прекращаются. Здесь Q_0 — модулярность перед обработкой текущего уровня, $\alpha, \delta \in (0, 1)$. При $Q_0 \geq 0$ принятый набор увеличивает модулярность.

Принятые перемещения расширяют поиск на соседей и затронутые сообщества. После уточнения внутри сообществ и обновления агрегированных весов и межуровневых связей поиск продолжается на следующем уровне. Перемещения и уточнение прекращаются после M итераций или при доле перемещенных среди просмотренных вершин меньше $\beta \in (0, 1)$. При исчерпании областей поиска или достижении N проходов разбиение переносится на исходный граф.

Веса следующего уровня вычисляются по формуле $W^{r+1}(x, y) = \sum_{u \in C_x, v \in C_y} W^r(u, v)$, где C_x, C_y — сообщества уточненного разбиения уровня r, представленные вершинами x, y, а W^r — веса уровня r. Пересчет вкладов затронутых ребер дает те же веса, что и полная агрегация.

При фиксированных N, L, M время обработки пакета на одном потоке составляет $O\big(|\Delta_t| + \sum_{i=1}^{N}\sum_{r=0}^{L}(e_{ri} + n_{ri}\log n_{ri})\big)$, где L — верхний уровень, n_{ri}, e_{ri} — числа различных просмотренных вершин и инцидентных им ребер на уровне r за проход i. Линейная часть учитывает просмотр инцидентных ребер и обновление агрегированных весов, а логарифмическая — сортировку не более одного перемещения на вершину за итерацию.

Иерархия и рабочие данные занимают $O((n+e)\ell + n + p)$ памяти, где $n = |V_t|$, $e = |E_t|$, $\ell = L + 1$ — число уровней, p — число потоков.

Экспериментальные результаты. Использованы графы KONECT и SNAP[1]: ca-cit-HepPh, youtube, com-lj, com-orkut, asia_osm, europe_osm, wikipedia, flickr, stackoverflow. Замеры проводились на одном потоке со 100 пакетами и $\gamma = 1$ на двухпроцессорном сервере (96 ядер, 512 ГиБ ОЗУ). Ребра временных графов сортировались по времени и перемешивались внутри пакетов; ребра статических графов перемешивались до разбиения на пакеты.

Leiden запускался на всем графе с предыдущим разбиением. GVE-Leiden и DF-Leiden вычислялись на первых шести неориентированных графах списка. В таблице приведены геометрические средние $T_{\text{base}}/T_{\text{LD}}$ и арифметические средние $Q_{\text{LD}}/Q_{\text{base}}$ по графам. Время T_{LD} включает все этапы обработки 100 пакетов, в том числе обновление иерархии; модулярность Q измерялась после обработки последнего пакета обновлений[2].

Базовый метод	Leiden	GVE-Leiden	DF-Leiden
$T_{\text{base}}/T_{\text{LD}}$	25,64	6,21	4,31
$Q_{\text{LD}}/Q_{\text{base}}$	0,996	1,061	1,027

На шести неориентированных графах из списка и крупных графах friendster, dimacs10-uk и kmer_V1r (68,4–214 млн вершин, 465 млн–3,3 млрд ребер) среднее арифметическое T_1/T_{64} составляет 5,22, где T_p — время обработки 100 пакетов LD-Leiden на p потоках.

Благодарности. Работа поддержана Минэкономразвития России (соглашение № 139-15-2025-012, идентификатор 000000C313925P4H0002).

Список литературы

[1] M. E. J. Newman, M. Girvan, “Finding and evaluating community structure in networks”, *Physical Review E*, **69**:2 (2004), 026113. DOI: 10.1103/PhysRevE.69.026113.

[1] https://konect.cc/, https://snap.stanford.edu/data/

[2] Код и конфигурации запусков: https://github.com/mpailab/ldleiden. Параметры запусков LD-Leiden: $N = 2$, $L = 4$, $M = 5$, $\alpha = 0{,}001$, $\beta = 0{,}01$, $\delta = 0{,}5$.

[2] V. A. Traag, L. Waltman, N. J. van Eck, “From Louvain to Leiden: guaranteeing well-connected communities”, *Scientific Reports*, **9** (2019), 5233. DOI: 10.1038/s41598-019-41695-z.

[3] S. Sahu, K. Kothapalli, D. S. Banerjee, “Fast Leiden Algorithm for Community Detection in Shared Memory Setting”, *Proceedings of the 53rd International Conference on Parallel Processing*, ACM, 2024, 11–20. DOI: 10.1145/3673038.3673146.

[4] S. Sahu, “Heuristic-based Dynamic Leiden Algorithm for Efficient Tracking of Communities on Evolving Graphs”, 2024, arXiv: 2410.15451. DOI: 10.48550/arXiv.2410.15451 .

Статья поступила 8 ноября 2026 г.

LD-Leiden: Local Parallel Community Detection in Dynamic Graphs

G. Bokov, A. Konovalov, A. Uporova, S. Moiseev, I. Safonov, A. Radionov

LD-Leiden locally updates the Leiden hierarchy and coordinates parallel vertex moves in dynamic weighted graphs. On nine real-world graphs (up to 50.9M vertices and 117M edges), geometric-mean speedup over repeated Leiden is 25.64, with a mean final modularity ratio of 0.996. On graphs up to 214M vertices and 3.30B edges, scaling from 1 to 64 threads gives a mean speedup of 5.22.

Keywords: dynamic graph, community detection, modularity, Leiden.

References

[1] M. E. J. Newman, M. Girvan, “Finding and evaluating community structure in networks”, *Physical Review E*, **69**:2 (2004), 026113. DOI: 10.1103/PhysRevE.69.026113.

[2] V. A. Traag, L. Waltman, N. J. van Eck, “From Louvain to Leiden: guaranteeing well-connected communities”, *Scientific Reports*, **9** (2019), 5233. DOI: 10.1038/s41598-019-41695-z.

[3] S. Sahu, K. Kothapalli, D. S. Banerjee, “Fast Leiden Algorithm for Community Detection in Shared Memory Setting”, *Proceedings of the 53rd International Conference on Parallel Processing*, ACM, 2024, 11–20. DOI: 10.1145/3673038.3673146.

[4] S. Sahu, “Heuristic-based Dynamic Leiden Algorithm for Efficient Tracking of Communities on Evolving Graphs”, 2024, arXiv: 2410.15451. DOI: 10.48550/arXiv.2410.15451.

Received on November 8, 2026