ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ АНАЛИЗА СЕТЕЙ НА ОСНОВЕ ОБХОДА ПУЧКА ТРАЕКТОРИЙ

Макрушин С.В., Панов Е. В.
Финансовый университет при Правительстве Российской Федерации, г. Москва
SVMakrushin@fa.ru

При решении современных прикладных задач анализа сетевых структур исследователям и индустриальным игрокам зачастую приходится иметь дело с сетями (графами) большого масштаба, имеющим десятки, сотни миллионов узлов и более и зачастую имеющими существенно более миллиарда связей. Примерами таких сетей могут быть крупные универсальные базы знаний [1, 2, 3], социальные сети [4] и веб-граф и его фрагменты.

Классическим подходом к решению задач обработки больших данных является создание массово-параллельных алгоритмов для горизонтально-масштабируемых систем (кластеров, GPGPU решений) [5]. Но важнейшим требованием для эффективной реализации массово-параллельных алгоритмов является локализация данных, что является проблемой для классических подходов к хранению сетевых структур. Обход узлов сети при выполнении целевого алгоритма требует перехода от текущего узла к связанному с ним узлу, обычно находящемуся в произвольной позиции в памяти.

Повысить локальность хранения узлов в памяти возможно за счет явного или неявного поиска решения задачи наименьшего k-разреза графа и локального хранения компонент, выделенных этим алгоритмом. На практике для больших сетей эта задача чрезвычайно сложна и может решаться только приближенно и с использованием дополнительных эвристик. Более того, из-за очень неравномерного распределения степеней узлов, типичного для эмпирических сетей, разделение графа на слабо связные подграфы будет приводить к локализации самой плотной части сети на небольшом количестве вычислительных узлов, что для большинства алгоритмов анализа сетей приведет к непропорциональной нагрузке на вычислительные узлы. Для решения задач анализа сетей большого масштаба было предложено несколько подходов и основанных на них инструментов. Самым известным из них является предложенная компанией Google программная модель Pregel [6]. Однако и они имеют свои слабые места, например в Pregel проблема нелокальности проявляется в виде интенсивного обмена сообщениями между всеми связанными узлами сети.

Предлагаемый алгоритм обхода пучка траекторий (ОПТ) по сути является адаптацией метода частиц для решения задач, основанных на случайном блуждании по сети. Известно, что метод частиц имеет большой потенциал для построения массово-параллельных алгоритмов, однако в случае его применения к сетям сохраняется проблема нелокальности представления сети в памяти. Для решения этой проблемы нами предложена специальная структура данных: множество

траекторий. По сути, каждая траектория представляет собой траекторию случайного блуждания по графу. Узлы траектории относятся к соответствующим узлам (вершинам) исходной сети, а каждой связи (ребру) исходной сети соответствует единственный переход (пара последовательных узлов) в одной из траекторий множества траекторий (см. рис. 1а и 1б).

Структура исходной сети в алгоритме ОПТ хранится только в неявном виде – в виде множества траекторий, из которого может быть однозначно восстановлена. Значения целевого алгоритма, которые нужно хранить в узлах в явном виде, не хранятся, а хранятся на траекториях, проходящих через него в виде набора «представлений» значений. Алгоритм обхода пучка траекторий не гарантирует совпадения значений представлений одного узла, но при этом стремится обеспечить конвергенцию значений представлений за счет массового выполнения операции попарного усреднения значений представлений, относящихся к соответствующим узлам.

В рамках алгоритма ОПТ принимается, что вычислительные ресурсы представляются в виде множества вычислительных узлов (например, узлов кластера или streaming multiprocessor в GPGPU архитектуре CUDA), на каждом из которых параллельно исполняется множество потоков. Синхронизация и обмен данными между потоками в рамках одного узла доступны с низкими издержками. Благодаря хранению сети в виде множества траекторий каждый вычислительный узел может работать с локализованным блоком данных (подмножеством траекторий) и осуществлять последовательный обход траекторий множеством потоков, параллельно работающих на каждом из вычислительных узлов. Множество траекторий, обрабатываемых одним вычислительным узлом, мы называем «пучком траекторий». Каждый поток последовательно обходит свою траекторию, выполняет на ней необходимые вычисления и с низкими издержками обменивается результатами с другими потоками в рамках пучка. Так как траектории хранятся в памяти последовательно, это исключает обращение к произвольным областями памяти и существенно ускоряет работу с оперативной памятью.

Для обеспечения глобальной конвергенции значений выполняется миграция траекторий между подмножествами, доступными каждому из вычислительных узлов. Так как от этого процесса не зависит непосредственный обход траекторий, высокая латентность и относительно низкая скорость обмена данными между узлами не является существенным ограничением для алгоритма. При этом существенным ограничением является то, что целевой аналитический алгоритм, который будет использовать инфраструктуру алгоритма ОПТ, должен базироваться на логике случайных блужданий. Примерами таких алгоритмов является одна из реализаций алгоритма PageRank [7] и некоторые варианты построения векторных представлений для узлов сетей [8, 9] и графов знаний [10].

Аналогично Pregel и другим алгоритмам, предоставляющим инфраструктуру для целевых алгоритмов анализ графов, алгоритм ОПТ требует, чтобы целевой алгоритм был сформулирован в

виде реализации нескольких функций, вызываемых в процессе реализации массивно-параллельного случайного блуждания по графу. В частности, должна быть реализована функция $U(\cdot, \cdot)$ изменения значений в очередном узле, посещаемом при случайном блуждании и функция $A(\cdot, \cdot)$ попарного усреднения значений приписываемых одному узлу, но хранимых в разных траекториях.

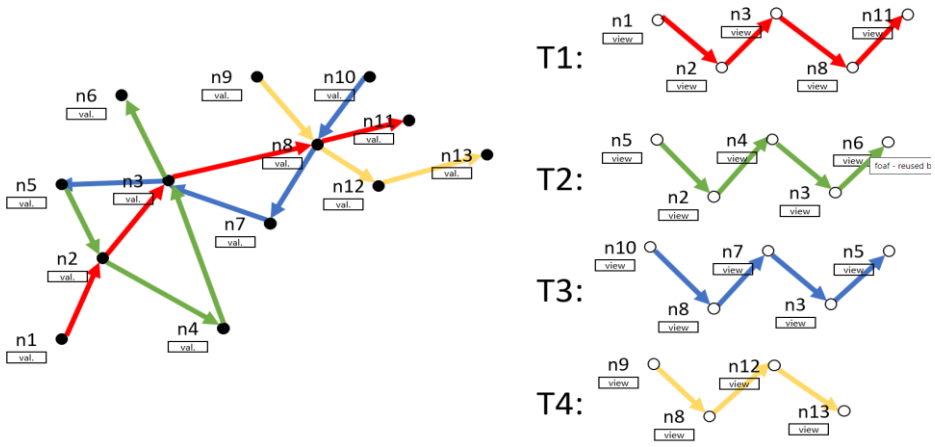
При реализации случайного блуждания по графу в алгоритме ОПТ реализовано четыре операции над траекториями: update, merge, exchange, read. При выполнении операции update осуществляется обход траектории и изменение значений для узлов с помощью функции $U(\cdot, \cdot)$. Повторный обход неизменной траектории с операцией update не целесообразен, т.к. это будет противоречить логике организации случайного блуждания. При выполнении функции merge осуществляется удлинение траекторий путем слияния двух траекторий, у которых совпадают хвостовой и головной узел. Значения для узла склейки усредняется функцией $A(\cdot, \cdot)$, а хвост удлиненной траектории подлежит обходу с помощью update (см. рис. 2а). Удлинение траекторий повышает эффективность работы алгоритма ОПТ и позволяет формировать описание сети в виде множества траекторий при первой обработке сети. Операция exchange позволяет «перемешивать» пары траекторий, в которых имеется совпадающий внутренний узел. В результате операции exchange при помощи функции $A(\cdot, \cdot)$ выполняется усреднение значений для совпадающего узла и формируются измененные траектории, для которых стартуют новые операции update (см. рис. 2б). Операции merge и exchange осуществляются только в рамках одного пучка траекторий, для повышения частоты этих операций в пучках используются вытесняющие кэши, хранящие информацию о последних узлах рассмотренных в пучке траекторий. Для пополнения кэшей целесообразно повторно обходить траектории, для которых выполнялся update, не изменяя при этом значения узлов. Эта операция названа read.

На данный момент ведется тестирование последовательного прототипа алгоритма ОПТ, реализованного на языке Python, после завершения реализации запланирована разработка параллельной версии алгоритма, а затем перенос на его одну из целевых платформ: кластер или GPGPU архитектуру CUDA.

Литература

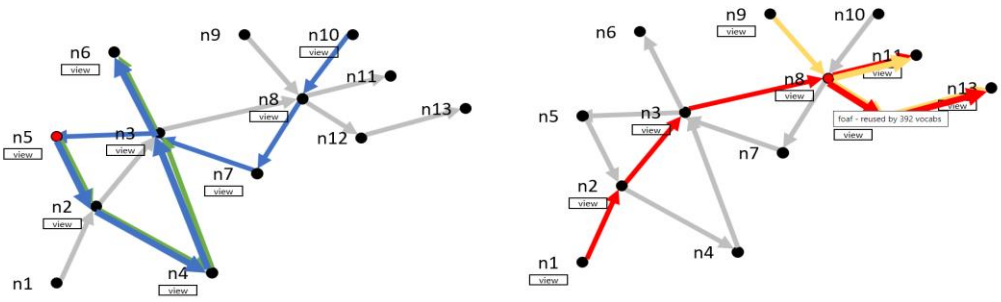
1. База знаний Wikidata [Электронный ресурс] – Режим доступа: https://www.wikidata.org, свободный (15.11.2019).
2. База знаний DBpedia [Электронный ресурс] – Режим доступа: https://wiki.dbpedia.org, свободный (15.11.2019).
3. Official Google Blog: Introducing the Knowledge Graph: things, not strings. [Электронный ресурс] – Режим доступа: https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html, свободный (15.11.2019).
4. Facebook API Graph. [Электронный ресурс] – Режим доступа: https://developers.facebook.com/docs/graph-api, свободный (15.11.2019).

5. *Jeffrey Dean, Sanjay Ghemawat*, Mapreduce: simplified data processing on large clusters // Commun. ACM, 51(1):107–113, 2008.
6. G. Malewicz, *et al*. Pregel: a system for large-scale graph processing // In SIGMOD, 2010
7. *L. Page, S. Brin, et al*. The PageRank Citation Ranking: Bringing Order to the Web // Technical report, Stanford Digital Library Technologies Project, 1998
8. *A. Grover, J. Leskovec*, node2vec: Scalable feature learning for networks // in: Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.
9. *B. Perozzi, R. Al-Rfou, S. Skiena*, Deepwalk: Online learning of social representations // in: Proceedings 20th international conference on Knowledge discovery and data mining, 2014, pp. 701–710.
10. *N. Lao, T. Mitchell, and W. W. Cohen,* Random walk inference and learning in a large scale knowledgebase // in Proceedings of the Conference on Empirical Methods in Natural Language Processing. ACL 2011, pp. 529–539.



а. Исходная сеть с ориентированными связями б. Множество траекторий

Рис. 1 Пример исходной сети и построенного на ее основе множества траекторий



а. Результат операции merge для траекторий T3 и T2
(для узла 5 проведено усреднение значений представлений)

б. Результат операции exchange для траекторий T1 и T4
(для узла 8 проведено усреднение значений представлений)

Рис. 2 Выполнение операций merge и exchange для траекторий рассмотренной сети