Улучшение качества разбиения графа с помощью многоуровневой оптимизации[1]

Р.К. Пастухов <pastukhov@ispras.ru>
А.В. Коршунов <korshunov@ispras.ru>
Д.Ю. Турдаков <turdakov@ispras.ru>
С.Д. Кузнецов <kuzloc@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25

Аннотация. Разбиение графа необходимо для решения задач, связанных с обработкой графов, данные которых распределены по нескольким дискам или вычислительным узлам. Эта задача хорошо изучена, но большинство ее решений не подходит для обработки графов с миллиардами вершин на вычислительных кластерах, т.к. эти решения предназначены для вычислительных машин с общей памятью либо для суперкомпьютеров с возможностью посылать сообщения с минимальными задержками. Один из подходов, позволяющий решать задачу разбиения графа на кластерах, – это метод Balanced Label Propagation, основанный на алгоритме распространения меток. В данной работе предлагается метод, позволяющий использовать многоуровневую оптимизацию для улучшения качества разбиений, получаемых с помощью алгоритма Balanced Label Propagation.

Ключевые слова: разбиение графов, распространение меток, многоуровневая оптимизация, социальные сети, распределенные вычисления.

1. Введение

Задача разбиения графа возникает в таких областях, как параллельные и распределенные вычисления, научные вычисления и проектирование электронных схем. Задача заключается в том, что требуется разбить множество вершин графа на k непересекающихся частей примерно одинакового размера так, чтобы количество ребер (или их суммарный вес, если граф взвешенный) с концами в разных частях было минимальным. Эта задача является NP-трудной даже при $k = 2$ [1], поэтому используются приближенные решения.

Задачу разбиения графа можно формализовать следующим образом: для данного графа $G = (V, E)$ нужно таким образом разбить множество вершин на

[1] Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта по гранту №13-07-12134

k частей $V_1, V_2, \dots, V_k$ таких, что $V_i \cap V_j = \emptyset$ при $i \neq j$, $\cup_i V_i = V$, $L_i \leq |V_i| \leq U_i$, чтобы минимизировать мощность разреза $W_{cut} = |\{(a,b) \in E: a \in V_i, b \in V_j, i \neq j\}|$. Параметры L_i и U_i задают ограничения на размеры частей. Часто ограничения для всех частей одинаковы, например, $L_i = \lfloor |V|/k \rfloor, U_i = \lceil |V|/k \rceil$. Если вершинам графа назначены веса, то вместо $|V_i|$ следует использовать $W(V_i) = \sum_{v \in V_i} w(v)$, где $w(v)$ – вес вершины v. Аналогично, если граф взвешен, то $W_{cut} = \sum_{e \in E_{cut}} w(e)$, где $E_{cut} = \{(a,b) \in E: a \in V_i, b \in V_j, i \neq j\}$, а $w(e)$ – вес ребра e. В случае с заданными весами вершин жесткие ограничения, задаваемые параметрами L_iи U_i, достаточно трудно удовлетворить (возникает задача, похожая на задачу о рюкзаке). Поэтому часто эти условия делают менее жесткими, добавляя коэффициент мягкости $f > 0$: $L_i = \lfloor (1-f)W(V)/k \rfloor, U_i = \lceil (1+f)W(V)/k \rceil$.

Такое разбиение также называют разрезом по ребрам. При распределенной обработке графов также используются разрезы по вершинам [12], когда на части разбивается не множество вершин, а множество ребер графа. Формально эта задача эквивалентна разрезу по ребрам реберного графа, но обычно для ее решения используют специализированные алгоритмы. Разрезы по вершинам в данной работе не рассматриваются.

Один из подходов [2], показывающий наилучшие результаты, основан на алгоритме Кернигана-Лина [4] и многоуровневой оптимизации. Этот алгоритм достаточно эффективен и позволяет находить хорошие разбиения, но не подходит для распределенных систем.

Для разбиения больших социальных графов был предложен алгоритм Balanced Label Propagation [3]. Он основан на алгоритме распространения меток из [5] и может использоваться в распределенных системах, поддерживающих парадигму Map-Reduce [6].

В следующем разделе кратко описываются алгоритм Balanced Label Propagation и суть метода многоуровневой оптимизации. Затем предлагается подход, в котором комбинируется применение Balanced Label Propagation и многоуровневой оптимизации. В четвертом разделе обсуждаются результаты экспериментов применения подхода авторов, реализованного в среда Sparc [7], к разбиению двух реальных графов социальных сетей. В заключении отмечаются преимущества предложенного подхода и формулируется ряд нерешенных исследовательских проблем.

2. Используемые алгоритмы и методы

2.1 Balanced Label Propagation

Алгоритм поиска сообществ [5], основанный на распространении меток, показал эффективность такого подхода в рамках распределенных систем. В нем каждой вершине присваивается уникальная метка, затем несколько раз производится следующая операция: метка каждой вершины заменяется на

наиболее часто встречающуюся метку среди ее соседей (если таких меток несколько, одна из них выбирается случайно). Таким образом, каждая вершина стремится попасть в группу вершин, в которой количество внутренних ребер, инцидентных вершине, будет минимально. При этом вычисления можно разбить на два этапа: генерация сообщений (меток) и получение сообщений. При этом до окончания этапа генерации доставка сообщений не требуется. Это позволяет эффективно реализовать данный алгоритм на основе парадигмы Map-Reduce.

Получаемое разбиение на сообщества не является решением задачи разбиения графа, потому что количество сообществ нельзя задать заранее, и их размеры не сбалансированы. Для обеспечения разбиения графа у алгоритма Balanced Label Propagation имеются два отличия от описанного алгоритма, основанного на распространении меток:

- при инициализации метки назначаются таким образом, чтобы ограничения по балансировке соблюдались (например, случайно);
- для изменения в соответствии с алгоритмом распространения меток выбирается только часть вершин графа таким образом, чтобы ограничения по балансировке при этом не были нарушены.

Выбор вершин, метки которых изменяются, происходит с помощью решения задачи линейного программирования, с целью минимизации количества ребер между частями при соблюдении ограничений по балансировке.

Алгоритм достаточно быстро сходится; для большинства графов достаточно около 10 итераций.

2.2 Многоуровневая оптимизация

Многие алгоритмы разбиения графа, в частности, алгоритм Кернигана-Лина и Balanced Label Propagation, основаны на улучшении существующих разбиений и по сути являются алгоритмами локальной оптимизации: они неплохо оптимизируют существующее разбиение, но не всегда могут его существенно изменить. Многоуровневая оптимизация позволяет локальным оптимизациям работать с более крупными частями графа путем стягивания ребер в вершины. Ее можно разделить на три этапа: огрубление графа, разбиение огрубленного графа и уточнение.

Во время огрубления из исходного графа G формируется последовательность графов $G = G_0, G_1 \dots G_n$ такая, что граф G_{i+1} получается за счет стягивания некоторых ребер в графе G_i, образующих паросочетание. Обычно множество стягиваемых ребер выбирается так, что они образуют максимальное паросочетание. При этом, даже в исходном графе G не было весов, в генерируемых графах G_i как вершины, так и ребра должны быть взвешены. Это связанно с тем, что при объединении нескольких ребер или вершин вес образуемого элемента графа должен равняться сумме весов исходных элементов. Если после стягивания ребер вершина v в графе G_i переходит в

вершину u в графе G_{i+1}, будем называть u образом v и обозначать это $u = I(v)$. У каждой вершины в графах $G_i, 0 \le i < n$ имеется ровно один образ.

После этого задача разбиения решается на графе G_n. Увеличивая n, количество вершин в графе G_n можно сделать сколь угодно малым. Поэтому при получении разбиения этого графа можно (но не обязательно) использовать более сложные алгоритмы, чем при улучшении разбиений во время перехода от G_{i+1} к G_i на следующем шаге.

На этапе уточнения разбиения огрубленных графов преобразуются в разбиения этих графов до огрубления. Чтобы получить разбиение графа G_i, имея разбиение графа G_{i+1}, достаточно разбить вершины в соответствии с разбиением их образов в G_{i+1}. То есть если набор V_i является разбиением графа G_{i+1}, то соответствующее ему разбиение G_i будет состоять из частей $V_i' = \{v: I(v) \in V_i\}$. Легко видеть, что у разбиения графа G_i веса частей и мощность разреза будут такими же, как и у разбиения G_{i+1}. После получения разбиения графа G_i оно улучшается с помощью используемого алгоритма локального улучшения разбиений. Эти действия повторяются, пока не получится разбиение графа $G_0 = G$.

3 Предлагаемый подход

Основная проблема, возникающая при распределенной многоуровневой оптимизации, состоит в сложности выбора пар вершин для стягивания на этапе огрубления графа. В [2] показано, что стягивание ребер с наибольшим весом позволяет улучшить качество разбиения по сравнению со стягиванием случайно выбранных ребер. В распределенной системе подобный подход сложно реализовать, поскольку данные о ребрах графа распределены по разным вычислительным узлам: один узел не может самостоятельно выбрать ребро для стягивания, так как другой вычислительный узел может выбрать смежное ему ребро, создавая конфликт (например, если разные узлы решат стягивать смежные ребра, стягиваемые ребра не будут образовывать паросочетание). С другой стороны, согласование выбора ребра со всеми вычислительными узлами, содержащими смежные ему ребра, приведет к существенной деградации производительности.

Достаточно простое решение – разбить вершины на группы (используя некоторое начальное разбиение графа, например, случайное) и выбирать для стягивания только такие ребра, чтобы обе инцидентные им вершины принадлежали одной группе – показывает хорошие результаты. Это позволяет производить стягивание ребер на каждом вычислительном узле независимо при условии, что памяти каждого вычислительного узла достаточно для обработки любой группы.

При этом может оказаться, что некоторые вершины не инцидентны никаким ребрам, целиком содержащимся в их группе. Это приводит к необходимости стягивать не только ребра, но и произвольные пары вершин без ребра между ними – доля таких вершин может быть достаточно большой, а для

эффективной работы желательно, чтобы количество вершин в графах G_i сокращалось экспоненциально.

Самый простой способ реализации такого подхода в распределенной системы, поддерживающей Map-Reduce, состоит в группировке вершин по идентификатору их группы в операции map с последующим стягиванием ребер внутри групп вершин в операции reduce. Оптимальное количество групп равно общему числу процессоров на всех машинах в кластере: меньшие значения не позволят использовать все ресурсы кластера, а при больших значениях будет расти количество вершин, изолированных в группах вершин, которые назначаются одному вычислительному узлу.

Если в начальном разбиении столько же частей, сколько в результирующем разбиении, то многоуровневую оптимизацию можно производить итеративно: результат разбиения, полученный с помощью многоуровневой оптимизации, можно использовать как начальное разбиение для следующей итерации. Теоретически это может улучшить качество разбиения, но подобные эксперименты в рамках данной работы не проводились.

Таким образом, предложенный метод состоит из следующих шагов.

1. Разбиение вершин исходного графа на группы. Для этого выбирается желаемое количество групп, и граф разбивается на подграфы с помощью небольшого количества итераций Balanced Label Propagation.
2. Огрубление графа. На этом шаге генерируется последовательность огрубленных графов (а также дополнительные данные, которые будут использоваться при уточнении). Сначала в каждой группе вершин выбираются ребра (или пары вершин) для стягивания. Результат выбора ребер для стягивания можно представить в виде отображения (набора пар) идентификаторов вершин в идентификаторы их образов в огрубленном графе. Затем создается огрубленный граф путем отождествления вершин, инцидентных стягиваемым ребрам. Детали этого этапа могут зависеть от возможностей используемой платформы распределенных вычислений. В общем случае достаточно заменить идентификаторы вершин на идентификаторы их образов и затем объединить одинаковые вершины и ребра, соединяющие одинаковые вершины. Имеется несколько способов реализовать это действие в с использованием Map-Reduce, но подходы, позволяющие избежать соединения данных графа с полученным отображением вершин на их образы, позволяют достигать лучшей производительности. Например, поддержка кеширования в системе Spark [7] позволяет сохранить группы вершин в памяти кластера, выбрать ребра для стягивания в пределах каждой группы, разослать эту информацию всем узлам кластера (здесь предполагается, что у каждого вычислительного узла в кластере достаточно памяти чтобы хранить информацию об образах всех вершин) и затем генерировать огрубленный граф, используя

данные из кеша.

3. Разбиение наименьшего из огрубленных графов с помощью алгоритма Balanced Label Propagation.
4. Уточнение графов. При этом после получения разбиения очередного уточненного графа (для этого достаточно обратить отображения вершин на их образы; это можно легко сделать как с помощью Map-Reduce, так и в памяти, если ее достаточно, чтобы хранить это отображение) оно улучшается с помощью нескольких итераций Balanced Label Propagation.
5. После получения в результате уточнения исходного графа с улучшенным разбиением вывести это разбиение как результат работы алгоритма.

4 Вычислительные эксперименты

Для тестирования алгоритм Balanced Label Propagation и его вариант, в котором используется многоуровневая оптимизация были реализованы с использованием системы Spark. Сравнение производилось на следующих наборах данных:

- граф связей в сети LiveJournal [8, 9]; содержащий 4.8 млн. вершин и 68.9 млн. ориентированных ребер; для тестирования граф был преобразован в неориентированный, количество ребер после этого составило 42.8 млн.;
- часть графа связей сети Facebook, полученная с помощью алгоритма MHRW [10, 11]; содержащая 0.9 млн. вершин и 1.79 млн. ребер.

В качестве меры качества разбиения использовалась доля ребер, обе вершины которых оказались в одном разделе. Если суммарный вес ребер графа (поскольку все графы в тестировании были без весов, он равен количеству ребер) W, а суммарный вес ребер в разрезе – W_{cut}, то доля внутренних ребер выражается формулой $(W - W_{cut})/W$. Видно, что чем меньше суммарный вес разреза, тем больше доля внутренних ребер. Но за счет нормализации этот параметр удобнее при сравнении результатов на различных графах.

При тестировании алгоритма, использующего многоуровневую оптимизацию, огрубление графа производилось до тех пор, пока количество вершин не станет меньше или равно 10000.

Перед началом огрубления вершины разбивались на разделы с помощью применения трех итераций Balanced Label Propagation к случайному разбиению. Для графа LiveJournal наилучшие результаты достигаются при 0..1 итерациях. Для части графа Facebook качество возрастает при увеличении количества итераций в интервале 0..10. Влияние начального разбиение на качество разбиения – это одно из возможных направлений дальнейших исследований.

После каждого перехода от меньшего графа к большему, разбиение оптимизировалось с помощью двух итераций Balanced Label Propagation. Увеличение количества итераций на этом этапе улучшает качество разбиения, но также увеличивает время работы.

На рис. 1 видно, что качество разбиения Balanced Label Propagation сходится достаточно быстро и после 10 итераций уже практически не меняется. В сравнительных тестах проводилось 20 итераций.

На рис. 2 показано, что многоуровневая оптимизация позволяет значительно улучшить качество разбиения при всех рассмотренных значениях количества разделов от 8 до 56.

Во всех тестах многоуровневая версия работала приблизительно в два раза дольше. Теоретически, многоуровневая версия может быть быстрее оригинальной, если оптимизировать процесс огрубления графов, но пока не ясно, насколько сильно можно улучшить производительность этого этапа. Также актуальной темой для дальнейших исследований является определение оптимального количества итераций Balanced Label Propagation на этапе уточнения графов и влияние их количества на производительность.

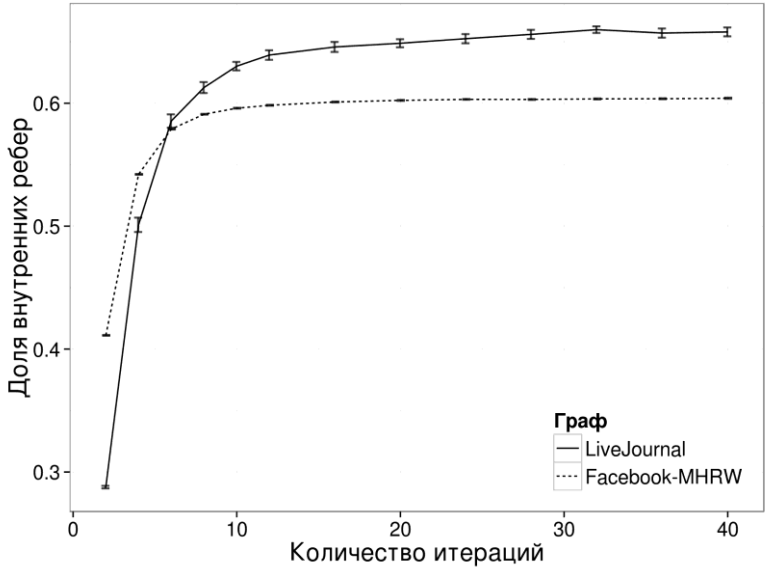

Рис. 1. Сходимость Balanced Label Propagation.

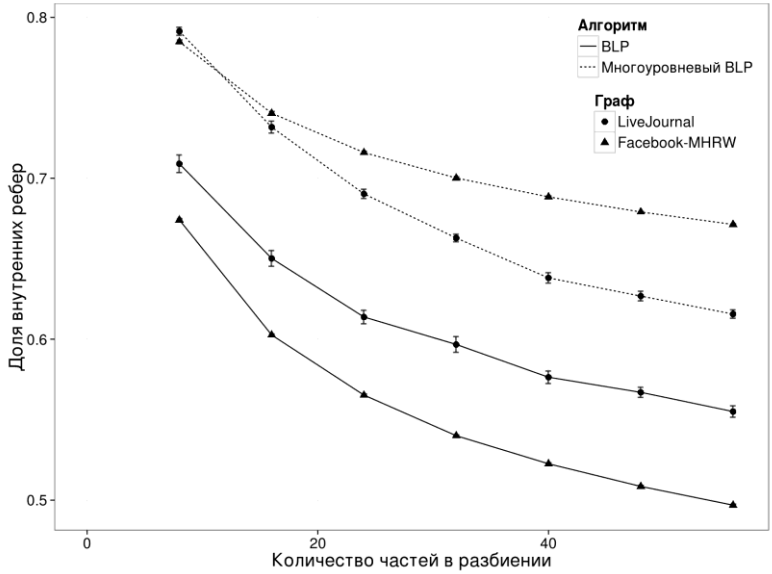

Рис. 2. Сравнение качества разбиений

5 Заключение

Было показано, что многоуровневая оптимизация позволяет заметно улучшить качество разбиений, получаемых с помощью алгоритма Balanced Label Propagation. Одна из выявленных проблем подхода связана с тем, что в исходном разбиении, используемом для генерации последовательности графов путем стягивания ребер, в каждом разделе может быть довольно много изолированных вершин, что может оказывать влияние на качество результирующего разбиения. Кроме того ребра для стягивания брались произвольно, хотя известно, что более сложные алгоритмы выбора ребер для стягивания могут улучшить качество разбиений [2]. Таким образом, актуальны следующие направления для дальнейших исследований:

- исследование возможного выигрыша от использования более сложных алгоритмов выбора ребер для стягивания в рамках одной группы вершин, например, алгоритма поиска паросочетаний большого веса из [2];
- исследование возможности распределенного стягивания ребер графа без ограничений на расположение вершин ребра относительно вычислительных узлов, например, за счет разрешения ситуаций, когда ребра не образуют паросочетания;
- исследование влияния начального разбиения на результаты, в том числе того, насколько лучших результатов можно достичь с помощью

итеративной многоуровневой оптимизации;

- улучшение производительности алгоритма.

Список литературы

[1]. M. R. Garey, D. S. Johnson, and L. Stockmeyer. “Some simplified NP-complete graph problems”. In: Theoretical computer science 1.3 (1976), pp. 237–267.

[2]. G. Karypis and V. Kumar. “A fast and high quality multilevel scheme for partitioning irregular graphs”. In: SIAM Journal on scientific Computing 20.1 (1998), pp. 359–392.

[3]. J. Ugander and L. Backstrom. “Balanced Label Propagation for Partitioning Massive Graphs”. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM ’13. Rome, Italy: ACM, 2013, pp. 507–516. isbn: 978-1-4503-1869-3. doi: 10.1145/2433396.2433461.

[4]. B. W. Kernighan and S. Lin. “An Efficient Heuristic Procedure for Partitioning Graphs”. In: Bell System Technical Journal 49.2 (1970), pp. 291–307. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[5]. U. Raghavan, R. Albert, and S. Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In: Physical Review E 76.3 (2007). doi: 10.1103/physreve.76.036106.

[6]. J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: OSDI 2004. 2004, pp. 137–150.

[7]. M. Zaharia et al. “Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing”. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI’12. San Jose, CA: USENIX Association, 2012.

[8]. Stanford Network Analysis Project, LiveJournal social network. 2006. url: http://snap.stanford.edu/data/soc-LiveJournal1.html.

[9]. L. Backstrom et al. “Group Formation in Large Social Networks: Membership, Growth, and Evolution”. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’06. Philadelphia, PA, USA: ACM, 2006, pp. 44–54. isbn: 1-59593-339-5. doi: 10.1145/1150402.1150412.

[10]. Sampling Online Social Networks, Facebook Social Graph. 2009. url: http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks#facebook_social_graph_-_mhrw_uni.

[11]. M. Gjoka et al. “Walking in Facebook: A Case Study of Unbiased Sampling of OSNs”. In: Proceedings of the 29th Conference on Information Communications. INFOCOM’10. San Diego, California, USA: IEEE Press, 2010, pp. 2498–2506. isbn: 978-1-4244-5836-3. doi: 10.1109/infcom.2010.5462078.

[12]. J. E. Gonzalez et al. “PowerGraph: Distributed Graph-parallel Computation on Natural Graphs”. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. OSDI’12. Hollywood, CA, USA: USENIX Association, 2012, pp. 17–30. isbn: 978-1-931971-96-6.

Improving quality of graph partitioning using multilevel optimization

R. Pastukhov <pastukhov@ispras.ru>
A. Korshunov <korshunov@ispras.ru>
D. Turdakov <turdakov@ispras.ru>
S. Kuznetsov <kuzloc@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. Graph partitioning is required for solving tasks on graphs that need to be split across disks or computers. This problem is well studied, but most results are not suitable for processing graphs with billons of nodes on commodity clusters, since they require shared memory or low-latency messaging. One approach suitable for cluster computing is Balanced Label Propagation, based on distributed label propagation algorithm for community detection. In this work we show how multi-level optimization can be used to improve partitioning quality of Balanced Label Propagation. One of major difficulties with distributed multi-level optimization is finding a matching in the graph. The matching is needed to choose pairs of vertices for collapsing in order to produce a smaller graph. As this work shows, simply splitting graph into several parts and finding matching in these parts independently is enough to improve the quality of partitioning generated by Balanced Label Propagation. Proposed algorithm can be implemented within any framework that supports MapReduce. In our experiments, when graphs were partitioned into 32 parts, ratio of edges that don't cross partitions increased from 54-60% to 66-70%. One of significant problems of our implementation is performance – work time of multi-level algorithm was approximately twice that of the original algorithm. It seems likely that implementation can be improved so that multi-level algorithm would achieve better computational performance as well as partitioning quality.

Keywords: graph partitioning, label propagation, multi-level optimization, social networks, cluster computing.

References

[1]. M. R. Garey, D. S. Johnson, and L. Stockmeyer. "Some simplified NP-complete graph problems". In: Theoretical computer science 1.3 (1976), pp. 237–267.

[2]. G. Karypis and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In: SIAM Journal on scientific Computing 20.1 (1998), pp. 359–392.

[3]. J. Ugander and L. Backstrom. "Balanced Label Propagation for Partitioning Massive Graphs". In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM '13. Rome, Italy: ACM, 2013, pp. 507–516. isbn: 978-1-4503-1869-3. doi: 10.1145/2433396.2433461.

[4]. B. W. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs". In: Bell System Technical Journal 49.2 (1970), pp. 291–307. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[5]. U. Raghavan, R. Albert, and S. Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In: Physical Review E 76.3 (2007). doi: 10.1103/physreve.76.036106.

[6]. J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: OSDI 2004. 2004, pp. 137–150.

[7]. M. Zaharia et al. “Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing”. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI’12. San Jose, CA: USENIX Association, 2012.

[8]. Stanford Network Analysis Project, LiveJournal social network. 2006. url: http://snap.stanford.edu/data/soc-LiveJournal1.html.

[9]. L. Backstrom et al. “Group Formation in Large Social Networks: Membership, Growth, and Evolution”. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’06. Philadelphia, PA, USA: ACM, 2006, pp. 44–54. isbn: 1-59593-339-5. doi: 10.1145/1150402.1150412.

[10]. Sampling Online Social Networks, Facebook Social Graph. 2009. url: http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks#facebook_social_graph_-_mhrw_uni.

[11]. M. Gjoka et al. “Walking in Facebook: A Case Study of Unbiased Sampling of OSNs”. In: Proceedings of the 29th Conference on Information Communications. INFOCOM’10. San Diego, California, USA: IEEE Press, 2010, pp. 2498–2506. isbn: 978-1-4244-5836-3. doi: 10.1109/infcom.2010.5462078.

[12]. J. E. Gonzalez et al. “PowerGraph: Distributed Graph-parallel Computation on Natural Graphs”. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. OSDI’12. Hollywood, CA, USA: USENIX Association, 2012, pp. 17–30. isbn: 978-1-931971-96-6.