Christopher Blackman

December 14, 2019

1 Literary Review

As we approach the limit of De Morgan's law, we come to an age in computing where sequential computing is limited by processing speed. Using RAM (Random Access Memory) as our base model of computation we create optimal sequential algorithms. However, in the age of big data, these sequential algorithms are not powerful enough to process all the data in reasonable time. Thus, we turn to parallel computing to where we have two models of computation which attempt to divide work. The first model assumes that all knowledge is known about the problem, then divides work base accordingly; this is known as the field of parallel computing. The second approach assumes that only local information is known about the problem, and through a series of steps, protocol definitions, and message passing an answer can be derived/approximated; this is known as the field of distributed computing. However, there are trade-offs to both approaches like : speed, run-time, correctness, optimality. Most algorithms take a trade-off between these two fields as computation gets further away from processors.

In this age being able to extract information form graphs becomes more prevalent in analyzation of web link graphs, social networks, and many other graphs types. One property that we can extract is graph connectivity. In graph theory graph connectivity is the minimum number of vertices, or elements that need to be removed to separate nodes into isolated sub-graphs [7]. A very similar definition of this is the minimum cut of a graph where a minimum cut is defined as the minimum partition of vertices, and edges into two disjoint sets. For simple unweighted graphs David R. Krager et al [8] created a randomized sequential algorithm based on properties preserved when edge contraction is applied. The papers by David R. Krage et al [9], and Barbara Geissmann et al. [10] further improve the algorithm by adding parallelisation. The downside of parallel algorithms introduced is that they are still based on the Parallel Random Access Memory (PRAM) model which can be used to prove theoretical bounds on algorithm. But the algorithms introduced disregarded hardware limitations where CPU cache is finite, and message passing to start up threads/processes are costly. It is noted that the paper in Barbara Geissmann et al. [10] attempts to improve cache misses by creating monotonic sequences to be read, however, the paper still relies heavily on parallelism of subroutines.

Another version of this problem comes from weighted undirected graphs. The goal is to find a cut where the sum of the edge weights are minimum. The Stoer-Wagner algorithm by Mechthid Stoer et al. [11] is a good sequential recursive algorithm for solving undirected weighted graphs with non-negative weights in $O(|V||E| + |V|^2 \log |V|)$ time. Usually the min-cut with respect to weighted undirected graphs is linked with the maximum flow problem as the minimum cut is often the bottleneck of the network [12,13].

A subfield of graph connectivity is connected components. Connected components can be divided into two categories: strongly connected components (SCC), and weakly connected components (WCC). A graph is said to be strongly connected if every vertex is reachable from every other vertex. Then a SCC is the same definition, but the maximal sub-graph that keeps this property. The SCCs of a directed graph form partitions of sub-graphs where they themselves are strongly connected. WCCs of a directed graph form a sub-graph for every pair in the sub-graph there exists a path, but not necessarily a bijective path.

One of the earliest algorithms introduced is Tarjan's algorithm which uses the properties of depth first search (DFS) to find SCC given by Robert Tarjan [16]. Later Edsger W. Dikstra came up with the path-based strong component algorithm seen in [17]. A simpler algorithm was also found after named Kosaraju's algorithm and published by Micha Sharir [17]. However, all of these algorithms abuse depth first search to find SCC, yet for parallel algorithms DFS is hard to paralyze. In fact, it was proved by John H. Reif that DFS is P-complete [19]. Therefore parallel algorithms do not focus on DFS when computing strongly connected components seen in Lisa K. Fleischer et al. [20], and Sungpack Hong et al. [15].

There is another version of Connected Components in undirected graphs. Unlike the directed version, if there is a path, then that path is a SCC since edges are bidirectional. Therefore the undirected version of SCC focuses on finding disjoint sub-graphs in the original graph, since each disjoint sub-graph is itself a SCC of the original graph. One may also relate this to finding WCC in a directed graph, since a path represents a bijective edge in undirected graphs. However, by doing so the WCC may contain a SCC in the original graph. A simple sequential algorithm can be done to find this where you take a DFS, or a breadth first search (BFS). Once you have exhausted your search, you select a non-traversed vertex to find the next SCC [21]. There have also been algorithms for dynamically changing graphs researched by Yossi Shiloach et al. [22].

As graphs increase in size, so does the amount of processing power required. In the field of SCC on undirected graphs there have been several subfields in which parallelization has taken affect. The first parallel algorithms dealing with SCC on undirected graphs deal with spanning forests [23,24,25,26,27,28,29,30,31,32,33,34,35,36,37]. The algorithms by Uzi Vishkin et al. [36,4], and Awerbuch et al. [37] work by combining vertices into trees, such that a constant number of vertices are deleted every iteration, but does not guarantee a constant fraction of edges. Thus they require $O(|E|log|V|)$ work. The random algorithm by Reif, and Phillips work by contracting vertices in the same component, which guarantees a constant number of vertices every iteration, however, it again does not guarantee a constant fraction of edges. Thus the expected work is $O(|E|log|V|)$.

Work-efficient poly-logarithmic-depth parallel connectivity algorithms have been designed in theory [38,39,40,41,42,43]. These algorithms are based on random edge sampling, or filtering edges. However, these have complicated structures which may be impractical to implement in practice.

There have also been CC algorithms focused in the field of distributed computing, where labels are propagate throughout nodes. A certain rule is imposed at each node describing when, and how to propagate a label. When no more labels are propagated the algorithm completes. Min-Label Propagation [3] algorithms generally have work done in $O(D|E|)$.

Another kind of algorithm tries to emulate a Breath First Search (BFS). Where agating in parallel from a single vertex until a single component is traversed. The downside is that, each component must be search in sequential order, thus what we obtain in lack of conflict resolution opposed to distributed algorithms, we also get serialization. Moreover, more algorithms have taken a 'bottom up' approach reducing work to sub-linear in $|E|$[2].

The algorithm which we will be focusing on is based on Shilovach-Vishkin[4] which has $O(\log n)$ parallelism, an uses a maximum of $2|E| + |V|$ processors. Shilovach-Vishkin represent their CC as a forest where each tree represents a connected component. They have two key phases *hook*, and *shortcut*. *Hook* is responsible for connecting processed components together where it hooks new nodes with their respective component, or connected two components together given the current knowledge know of the graph. They prove that this process results in tree that have height at most $\lfloor \log_{3/2} n \rfloor + 2$. The *shortcut* phase results in a compression of the trees in the forest such that they have depth at most one.

2 Bibliography

[1]https://ieeexplore-ieee-org.proxy.library.carleton.ca/document/8425156/references#refere
[2]https://ieeexplore-ieee-org.proxy.library.carleton.ca/document/6877288/references#refere
[3]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=2733089
[4]https://www-sciencedirect-com.proxy.library.carleton.ca/science/article/pii/019667748290
[5]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=2612692

[6]https://link.springer.com/chapter/10.1007/3-540-45591-4_68
[7]http://diestel-graph-theory.com/index.html
[8]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=234534
[9]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=331608
[10]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=3210393
[11]https://dl.acm.org/citation.cfm?id=263872
[12]https://link.springer.com/chapter/10.1007/978-0-8176-4842-8_15
[13]https://dl.acm.org/citation.cfm?id=1942094
[14]https://dl.acm.org/citation.cfm?doid=362248.362272
[15]https://dl.acm.org/citation.cfm?id=2503246
[16]https://ieeexplore.ieee.org/document/4569669
[17]https://dl.acm.org/citation.cfm?id=550359
[18]https://www.sciencedirect.com/science/article/pii/0898122181900080
[19]https://www.sciencedirect.com/science/article/pii/0020019085900249
[20]https://dl.acm.org/citation.cfm?id=663154
[21]https://dl.acm.org/citation.cfm?id=362272
[22]https://dl.acm.org/citation.cfm?id=322235
[23]https://www.sciencedirect.com/science/article/pii/0020019082901314
[24]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=739745
[25]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=305744
[26]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=254195
[27]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=182530
[28]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=359141
[29]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=214077
[30]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=106163
[31]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=203622
[32]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=358650
[33]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=72952
[34]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=586952
[35]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=1398907
[36]https://www.sciencedirect.com/science/article/pii/0166218X84900192
[37]B. Awerbuch and Y. Shiloach. New connectivity and MSF algorithms for Ultracomputer and
[38]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=237563
[39]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=123143
[40]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=247524
[41]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=377897
[42]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=586952
[43]https://dl-acm-org.proxy.library.carleton.ca/citation.cfm?id=686427
[44]S. Hambrusch and L. TeWinkel. A study of connected component labeling algorithms on the
[45]S. Goddard, S. Kumar, and J. F. Prins. Connected components algorithms for mesh-connect
[46]https://www.ams.org/journals/proc/1956-007-01/S0002-9939-1956-0078686-7/home.html
[47]https://arxiv.org/abs/1605.06643