The full text is available at: http://dx.doi.org/10.1561/1000000044

Fast Uncovering of Graph Communities on a Chip: Toward Scalable Community Detection on Multicore and Manycore Platforms

The full text is available at: http://dx.doi.org/10.1561/1000000044

Fast Uncovering of Graph Communities on a Chip: Toward Scalable Community Detection on Multicore and Manycore Platforms

Ananth Kalyanaraman
School of EECS, WSU, Pullman, WA USA

Mahantesh Halappanavar
PNNL, Richland, WA USA

Daniel Chavarría-Miranda
PNNL, Richland, WA USA

Hao Lu
School of EECS, WSU, Pullman, WA USA

Karthi Duraisamy
School of EECS, WSU, Pullman, WA USA

Partha Pratim Pande
School of EECS, WSU, Pullman, WA USA

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

A. Kalyanaraman et al.. *Fast Uncovering of Graph Communities on a Chip: Toward Scalable Community Detection on Multicore and Manycore Platforms.* Foundations and Trends® in Electronic Design Automation, vol. 10, no. 3, pp. 145–247, 2016.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-132-0

Foundations and Trends® in Electronic Design Automation
Volume 10, Issue 3, 2016

Editorial Board

The full text is available at: http://dx.doi.org/10.1561/1000000044

Editorial Scope

Topics

Foundations and Trends® in Electronic Design Automation publishes survey and tutorial articles in the following topics:

- System level design
- Behavioral synthesis
- Logic design
- Verification
- Test
- Physical design
- Circuit level design
- Reconfigurable systems
- Analog design
- Embedded software and parallel programming
- Multicore, GPU, FPGA, and heterogeneous systems
- Distributed, networked embedded systems
- Real-time and cyberphysical systems

Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2016, Volume 10, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in Electronic Design Automation
Vol. 10, No. 3 (2016) 145–247

DOI: 10.1561/1000000044

now
the essence of knowledge

Fast Uncovering of Graph Communities on a Chip: Toward Scalable Community Detection on Multicore and Manycore Platforms

Ananth Kalyanaraman
School of EECS, WSU, Pullman, WA USA
ananth@eecs.wsu.edu

Mahantesh Halappanavar
PNNL, Richland, WA USA
hala@pnnl.gov

Daniel Chavarría-Miranda
PNNL, Richland, WA USA
daniel.chavarria@pnnl.gov

Hao Lu
School of EECS, WSU, Pullman, WA USA
luhowardmark@wsu.edu

Karthi Duraisamy
School of EECS, WSU, Pullman, WA USA
kduraisa@eecs.wsu.edu

Partha Pratim Pande
School of EECS, WSU, Pullman, WA USA
pande@eecs.wsu.edu

Contents

Abstract

Graph representations are pervasive in scientific and social computing. They serve as vital tools to model the interplay among different interacting entities.

In this paper, we visit the problem of community detection, which is one of the most widely used graph operations toward scientific discovery. Community detection refers to the process of identifying tightly-knit subgroups of vertices in a large graph. These sub-groups (or communities) represent vertices that are tied together through common structure or function. Identification of communities could help in understanding the modular organization of complex networks. However, owing to large data sizes and high computational costs, performing community detection at scale has become increasingly challenging.

Here, we present a detailed review and analysis of some of the leading computational methods and implementations developed for executing community detection on modern day multicore and manycore architectures. Our goals are to: a) define the problem of community detection and highlight its scientific significance; b) relate to challenges in parallelizing the operation on modern day architectures; c) provide a detailed report and logical organization of the approaches that have been designed for various architectures; and d) finally, provide insights into the strengths and suitability of different architectures for community detection, and a preview into the future trends of the area. It is our hope that this detailed treatment of community detection on parallel architectures can serve as an exemplar study for extending the application of modern day multicore and manycore architectures to other complex graph applications.

A. Kalyanaraman, M. Halappanavar, D. Chavarría-Miranda, H. Lu, K. Duraisamy, and P. P. Pande. *Fast Uncovering of Graph Communities on a Chip: Toward Scalable Community Detection on Multicore and Manycore Platforms*. Foundations and Trends® in Electronic Design Automation, vol. 10, no. 3, pp. 145–247, 2016. DOI: 10.1561/1000000044.

1

Graphs and Community Detection

Graphs are powerful representations. They serve as vital tools in modeling the interplay between different interacting components of naturally built and human-engineered systems. Today, numerous graph operations are routinely defined to mine for higher order structural information that is preserved in the network of interacting components. Some of these operations are simpler to compute – such as those relating to specific nodal queries (e.g., degrees, neighborhood search), graph traversals (breadth first search, depth first search), and detection of connected components. Other operations are more complex to compute and yet very useful as discovery tools. Belonging to the latter category is the *community detection* operation.

Simply put, community detection refers to the process of identifying tightly-knit subgroups of vertices in a large graph. These sub-groups, also referred to as "communities", represent vertices that are tied together by a strong relationship (on the basis of edge links) but not as strongly to vertices outside the community. In other words, the edge-based connectivity within a community simply outweighs the community's connection to the rest of the network. A key factor that complicates the detection of communities is the lack of any prior knowledge on the number of such communities or their sizes.

The presence of communities in a graph suggests a *modular* organization of its entities, in which smaller groups, identified as communities, interact cohesively within, while only weak interactions (if any) are observed across communities. For this reason, communities are also sometimes referred to as "*modules*".

Community detection is used in a broad range of applications, as a discovery tool to reveal hidden structures within real world networks that are often missed by other standard graph mining tools. For instance, an analysis of a social networking site such as Facebook could reveal virtual communities of friends and acquaintances, or of people who share similar interests. On the other hand, an analysis of a microblogging site such as Twitter could reveal a more dynamic and evolving set of virtual communities, as users tweet about different themes and topics. Searching for community structure is also useful in naturally built systems. For instance, it is now known from the analysis of protein interaction networks, that proteins within numerous biological systems interact with one another as modules.

Despite its high potential for discovery and advancement of our understanding of real world networks, the modern day application of community detection is challenged by its high computational cost, which is further exacerbated by the rapid growth in data sizes. Under various formulations, the problem of community detection is known to be NP-Hard (Brandes et al., 2008). Therefore, heuristics are used in practice to allow the processing of large inputs. However, even such heuristics could take very long time or may run out of memory on modern day inputs. For instance, the Internet contained ∼46 billion webpages as of August 2015, as indexed by Google[1]. Figure 1.1 is a pictorial depiction of the Internet.

As of March 2015, the social networking site Facebook, Inc. contained 1.18 billion active users. Note that the above sizes are in the number of nodes. The number of edges typically exceed the number of nodes by orders of magnitude.

Consequently, high performance computing (HPC) has a critical role to play in extending the application of community detection to

[1]See (`http://www.worldwidewebsize.com/`).

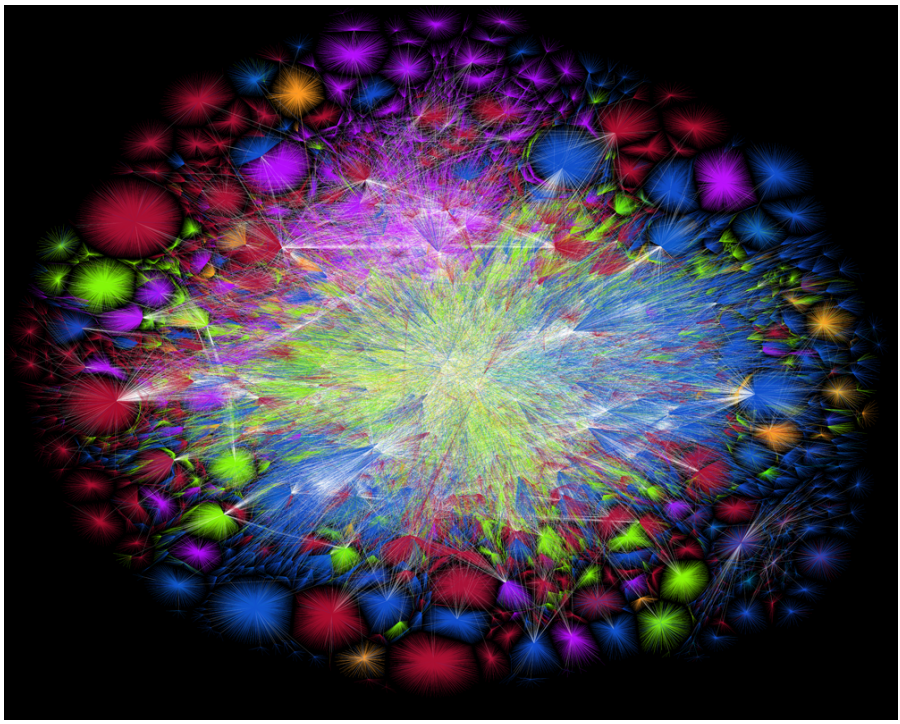

Figure 1.1: A map of the Internet as of July 2015, as reported by "The Opte Project / Barrett Lyon" (`http://www.opte.org/`). Colors show communities of internet address by their geographical region. The image has been reproduced with permission from the OPTE project (© 2014 by LyonLabs, LLC and Barrett Lyon).

modern day inputs. However, the application of HPC is not trivial. Community detection, as do most advanced graph operations, generates highly irregular computational and data access patterns that pose serious technical difficulties and scalability challenges during parallelization. This is in contrast to most classical scientific computing operations (e.g., Fast Fourier Transform, Molecular Dynamics simulations) where computations are largely regular and floating point calculations dominate the landscape.

The combinatorial nature of exploration within graph operations calls for newer paradigms in parallel algorithm and parallel architecture design. This specialized demand has generated a significant amount of

interest in graph applications among the HPC community, making the design and development of frameworks for optimized computation of complex graph operations such as community detection, a hotly researched topic.

In this paper, we have attempted to provide a comprehensive survey of this topic in what has become a continuously changing landscape in research. We focus on community detection and on how to achieve performance at scale for this operation using state-of-the-art multicore and many-core architectures. Our goal is to enlighten the reader on the following:

- the computational fundamentals and scientific underpinnings of the community detection operation (Chapters 1 and 2);
- classical approaches that have been developed for performing community detection (Chapter 3);
- details of two architectures as representatives of state-of-the-art multicore and manycore platforms (Chapter 4), and insights into the key algorithmic design challenges in parallelizing community detection on such architectures (Chapter 5);
- a detailed report and logical organization of parallel algorithms and architectures that have been designed for performing fast and scalable community detection on real world graphs (Chapter 6), along with results categorized by the target architecture (Chapter 7); and
- emerging Network-on-Chip (NoC) architectures and the improvements over modern day System-on-Chip architectures that they have to offer (Chapter 8).

More specifically, in Chapter 6, we cover parallelization in detail on traditional multicore systems (e.g., x86) and System-on-Chip (SoC) manycore architectures (specifically, EZchip Tilera TileGx). In Chapter 8, we cover parallelization on an emerging paradigm in NoC architectures that use wireless and wired on-chip interconnect links, and

report results through simulation studies. These emerging architectures offer a potential improvement over current SoC architectures.

Chapter 9 concludes the article with a discussion of the key results and insights, and a preview on future trends.

We hope that the aforementioned organization of the paper and its contents will expose an interested reader to key advances in parallel algorithms and architectures for community detection, and will eventually motivate the reader to contribute to future advances. We note that graph-theoretic operations happen to be a relatively new venue for exploration in many of these architectures, making it a vibrant area for research and development.

We note here that a complete survey on the broader topic of community detection is out of scope for this article. For that, the reader is referred to several authoritative survey articles that are already available (Fortunato, 2010; Papadopoulos et al., 2012). The material presented in this paper will introduce the community detection problem with all the necessary background. However, as stated above, the focus will rest on scalability and high performance computing methods and architectures designed for this graph operation. To the best of our knowledge, a survey article with a similar focus *does not* exist in the literature.

2

Community Detection: Background and Problem Definition

2.1 A Brief History of Networks and Community Detection

The documented origins of graph theory date back to 1736 when Leonhard Euler (1707-1783), a Swiss mathematician, was presented a problem by the city authorities of Königsberg, Prussia (Kaliningrad in modern day Russia). In what was to become known as the "Seven Bridges of Königsberg" problem, the city wanted to know whether it would be possible to parade through the town, crossing each of the seven bridges that connected the four land masses across the river Pregel, exactly once while returning to the origin of the circuit — as illustrated in Figure 2.1. Euler formulated this as a graph problem, by representing the land masses as "nodes" and bridges as "edges", and in fact arrived at a negative result for this circuit problem.

Despite the early work by Euler, it was not until the early part of the 20^{th} century when graph theory, as a field, started to take shape. Structural properties of networks conceived out of real world data started to emerge at this time. Scottish epidemiologists Kermack and McKendrick (1927) (Kermack and McKendrick, 1927) modeled the spread of infectious diseases as a network and proposed a theory to capture the rates

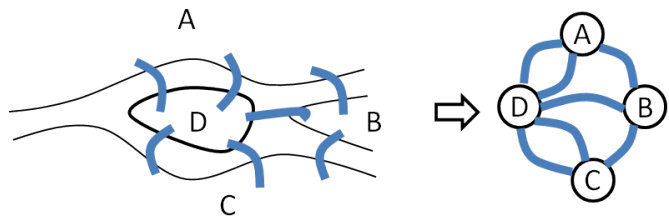

Figure 2.1: An illustration of Euler's Seven Bridges of Königsberg problem.

at which infection spreads and eventually terminates among a population. Rapoport and Solomonoff (1951) (Solomonoff and Rapoport, 1951) argued that naturally built systems such as neural nets and phenomena such as disease epidemiology can be represented in the form of random graphs. This study resulted in a systemic formulation and treatment of *random networks*, which would subsequently influence the later work by Paul Erdős and Alfréd Rényi in their pioneering work on Erdős-Rényi model (Erdös and Rényi, 1960). According to the Erdős-Rényi model, all possible graphs for a given set of vertices and a fixed number of edges are equally likely. In an equivalent random model, each edge has an independent and fixed probability of existence.

However, a significant departure from the random graph model was already observed by British statistician Udny Yule in 1927. In an attempt to derive a mathematical explanation for Darvin's evolutionary theory (Yule, 1925), he arrived at a notion that we today refer to as *preferential attachment*. He observed that the growth rates of individual genera (counted in the number of species within a genus) correlate with their previous sizes. This notion of preferential attachment, where the "rich" get "richer", would later be studied extensively in the context of degree distribution of real world networks, and in effect laying the foundation for *scale-free networks* (Barabási et al., 2000).

A second influential study in the context of non-random networks arose in the context of social sciences. Using a simple mailing experiment, Stanley Milgram at Yale (1967) concluded that the social network that connected people has a "six degree of separation" (Milgram, 1967).

This study pointed to surprisingly short pairwise distances in large social networks, and eventually this result would lead to the extensive study of *small-world networks* (Watts and Strogatz, 1998).

Despite these early advances, it was not until the 70′s that a clearer picture of the structural organization of non-random real world networks emerged. In a pivotal 1973 paper titled "The strength of weak ties" (Granovetter, 1973), Granovetter argued that social networks (aka. "sociograms") contained strongly tied groups (or "*communities*") built out of friendships and relationships, and weaker ties built out of acquaintances that acted as "bridges" between communities. This key observation was one of the first studies to reveal a higher order organization of networks into tightly-knit communities and their interconnection. Since then, such community-level organization has been discovered and studied in a wide variety of real world networks, both within nature-built and human-built systems. These include (but not limited to) the Internet, social networks such as Facebook and Twitter (Papadopoulos et al., 2012), *C. Elegans* neural system (Watts and Strogatz, 1998), protein interaction networks (Girvan and Newman, 2002), electric power grid (Newman, 2003), dolphin communication network (Connor et al., 1999), collaboration networks (Newman, 2003), customer preference databases (Reddy et al., 2002), and climate variability networks (Steinhaeuser et al., 2011). In these systems, discovering the community structures within networks have proved to be a critical step in advancing the knowledge and understanding of the underlying system and its functions.

2.2 Problem Definition

2.2.1 Basic Notation

Let $G(V, E, \omega)$ be an undirected weighted graph, where V is the set of vertices, E is the set of edges and $\omega(.)$ is a weighting function that maps every edge in E to a non-zero, positive weight. If the graph is unweighted, then we treat every edge to be of unit weight. In the input graph, edges that connect a vertex to itself are allowed — i.e., (i, i) can be a valid edge. However, multi-edges are not allowed. Let the

adjacency list of i be denoted by $\Gamma(i) = \{j|(i,j) \in E\}$. Let k_i denote the weighted degree of vertex i — i.e., $k_i = \sum_{j \in \Gamma(i)} \omega(i,j)$. We will use n to denote the number of vertices in G, m to denote the *number* of edges in the graph, and M to denote the sum of all edge weights — i.e., $M = \frac{1}{2} \sum_{i \in V} k_i$.

2.2.2 Problem Formulation

The generic version of the community detection problem is defined as follows:

Given an input graph $G(V, E, \omega)$, the objective is to compute a set of "communities" $C = \{c_1, c_2, \ldots c_k\}$. Each *community* c_i represents a maximal subset of V with a significantly higher concentration of edges flowing between its constituent vertices than to vertices in $V \setminus c_i$. We assume that the number of communities (k) and the sizes of the individual communities are *not* known *a priori*.

There are two variants of the community detection problem, based on whether to allow for overlaps between the computed communities or not. Many application use-cases enforce a partitioning (i.e., the output communities are mutually disjoint), whereas some application use-cases justify a relaxation to allow for potential overlaps among the output communities. Figure 2.2 shows an example for the partition version of the community detection problem.

2.2.3 Related Problems

It is noteworthy that community detection is not entirely unique in its goals and that there are other graph operations that are similar in their intent. Therefore, it is perhaps prudent to clarify on their relationships and differences.

Community detection is not to be confused with *clique finding*. In graph theory, cliques refer to complete subgraphs (i.e., each vertex has an edge to every other vertex in the subgraph). In community detection, this condition is relaxed. In other words, each vertex in a community is expected to be connected to most (if not all) vertices of the same community, and very few (if at all) vertices outside the community.

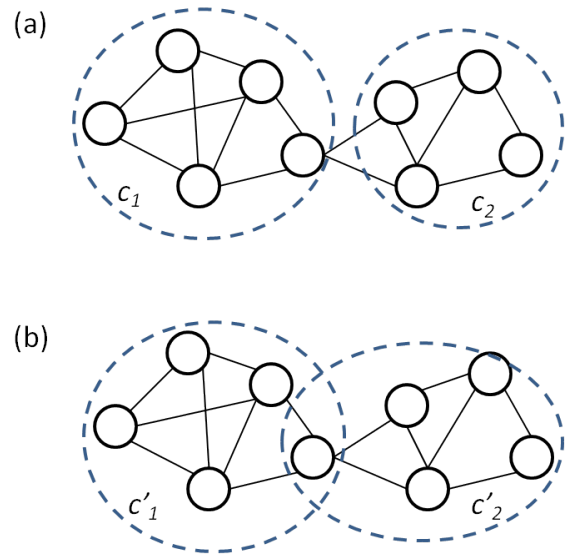

Figure 2.2: An example showing an input graph $G(V, E)$. Part (a) shows a candidate partitioning of its vertices into communities. Part (b) shows an alternative community structure that allows for overlaps between communities.

This relaxation to the clique criterion is essential for ensuring practical relevance. In real world, there are typically missing edges within otherwise strongly knit groups, and in some cases identifying remote homologs (related entities) become important.

Data clustering encompasses a broad class of algorithms that attempt to achieve a similar goal with grouping — except that the term "clustering" is used in a more generic context for clustering points of an arbitrary data type, over which a pairwise distance/similarity function is defined. It is possible to view community detection as a clustering problem applied to graphs — i.e., with the points represented by vertices and their pairwise relationships represented by edges. For this reason, the term community detection is sometimes used synonymously with *graph clustering*. However, the problem settings are still different. For instance, in clustering it is typical for the pairwise distance functions to respect the triangular inequality property. This need not be the case with the edge weights of a graph. Also, since graphs are defined in

a coordinate-free manner, it is typically not possible to take advantage of spatial data structures during implementation.

In graph-theory, there is another closely related problem that is similar to that of community detection. *Graph partitioning* is also a problem of binning vertices into groups. However, there are a few fundamental differences between the two problems. In graph partitioning, the number of partitions (i.e., clusters) and their expected sizes are known *a priori*. This is in contrast to community detection. In fact, the very purpose of community detection is to reveal the presence of any such community-wise organization of the input graph. On the other hand, graph partitioning aims to partition an input set of dependent tasks on a parallel system in a way that minimizes the cut (or dependencies across the partitions), while ensuring even load distribution. Consequently, graph partitioning is used as a load balancing tool while community detection is used as a discovery tool.

2.2.4 Measures for Community Detection

A critical aspect within the problem definition for community detection is how to measure the goodness of a given community assignment — i.e., the basis for binning vertices into the same community. Once such a measure is defined, it can be used as the optimization objective for computing communities.

Notable measures defined in the literature for evaluating the goodness of a community-wise partitioning include normalized cut (Shi and Malik, 2000), conductance (Kannan et al., 2004) and modularity (Newman, 2004b). An elaborate discussion on all their individual definitions is out of scope for this article, and a curious reader is referred to some of the authoritative reviews in literature (Fortunato, 2010; Papadopoulos et al., 2012).

In what follows, we select modularity for further discussion, as this measure has found a wide adoption in practice, and numerous algorithms and parallel implementations have been designed to compute a clustering based on modularity as the primary objective for optimization.

Modularity

Modularity was introduced by Mark E.J. Newman in his seminal paper (Newman, 2004b). It is a statistical measure that is defined for a given partitioning of an input graph (i.e., vertex to community assignment).

Given a graph $G(V, E, \omega)$, let $C = \{c_1, c_2, \ldots c_k\}$ denote a set of communities that represents a partitioning of the vertices in V. We will use $c(i)$ to denote the community containing vertex i. A community consisting of a single vertex is referred to as a *singleton*.

Note that the partitioning C also partitions the set of edges in G into two types:

- An *intra-community edge* is one that connects two vertices of the same community; and
- An *inter-community edge* is one that connects vertices of different communities.

Let $E_{i \to c}$ refer to the set of all edges connecting a vertex i to the vertices in a community c. And let $e_{i \to c}$ denote the sum of the edge weights for the edges in $E_{i \to c}$.

$$e_{i \to c} = \sum_{(i,j) \in E_{i \to c}} \omega(i, j) \tag{2.1}$$

Let a_c denote the sum of the weights of all edges incident on all vertices of community c (also referred to as *community degree*).

$$a_c = \sum_{i \in c} k_i \tag{2.2}$$

Note that in the above definition, each intra-community edge in community C is double-counted.

Definition 2.1. Given an input graph $G(V, E, \omega)$ and a community-wise partitioning of its vertices C, the **modularity** (denoted by Q) of the partitioning C is given by (Newman, 2004b):

$$Q = \frac{1}{2M} \sum_{i \in V} e_{i \to c(i)} - \sum_{c \in C} \left(\frac{a_c}{2M} \cdot \frac{a_c}{2M} \right) \tag{2.3}$$

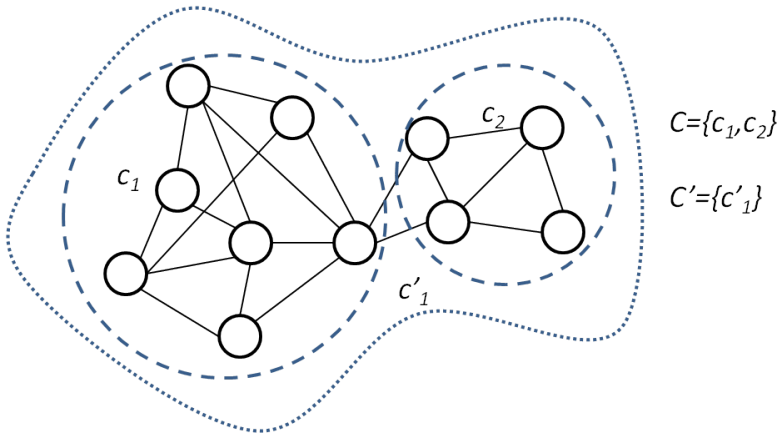

Figure 2.3: An example showing a simple graph input and two different partitions of its vertices (C and C'). The modularity of the partitioning C is given by $Q_C = 0.875 (= \frac{5+13}{20} - \frac{28+2}{40\times40})$. The modularity of the partitioning C' is given by $Q_{C'} = 0 (= \frac{20}{20} - \frac{40\times40}{40\times40})$.

The first term is the observed fraction of intra-community edges imposed by the partitioning C on V. The second term is a probabilistic estimate on the same fraction in a random graph with an identical vertex degree sequence. Intuitively, a good community-wise partitioning maximizes the fraction of intra-community edges. However, the fraction can be maximized by simply assigning all vertices to a single community. The second term is a statistical term that is intended to prevent such noisy outputs.

It should be easy to see that modularity is bounded by 1, and that a partitioning that achieves a value closer to 1 should be preferred. In other words, the community detection problem becomes one of computing a partition C that maximizes modularity.

Modularity optimization is an NP-Complete (Brandes et al., 2008).

Figure 2.3 shows a simple example of a graph and two different partitions of it alongside their modularity scores.

It is to be said that modularity is not necessarily the ideal metric for community detection. Issues such as resolution limit have been identified Fortunato (2010); Traag et al. (2011), and there are a few variants

of modularity definitions have also been devised Traag et al. (2011); Bader and McCloskey (2009); Berry et al. (2011). However, the definition provided in Eqn. (2.3) continues to be the more widely adopted version in practice, including for most of the methods for which parallelization has been implemented. Therefore, for sake of consistency with current literature, we stay with the aforementioned classical form of the definition.

References

Anant Agarwal, Liewei Bao, John Brown, Bruce Edwards, Matt Mattina, Chyi-Chang Miao, Carl Ramey, and David Wentzlaff. Tile processor: Embedded multicore for networking and multimedia. *Hot Chips*, 19, 2007.

Jason Ansel, Shoaib Kamil, Kalyan Veeramachaneni, Jonathan Ragan-Kelley, Jeffrey Bosboom, Una-May O'Reilly, and Saman Amarasinghe. Opentuner: An extensible framework for program autotuning. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, pages 303–316. ACM, 2014.

David A. Bader and Joe McCloskey. Modularity and graph algorithms. *SIAM AN10 Minisymposium on Analyzing Massive Real-World Graphs*, pages 12–16, 2009.

David A. Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. Graph partitioning and graph clustering. In *10th DIMACS Implementation Challenge Workshop*, 2012.

Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1):69–77, 2000.

Jonathan W. Berry, Bruce Hendrickson, Randall A. LaViolette, and Cynthia A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Physical Review E*, 83(5):056119, 2011.

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, page P10008, 2008.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

Umit Catalyurek, John Feo, Assefaw H Gebremedhin, Mahantesh Halappanavar, and Alex Pothen. Graph coloring algorithms for multi-core and massively multithreaded architectures. *Parallel Computing*, 38(11):576–594, 2012.

Daniel Chavarría-Miranda, Ajay Panyala, Mahantesh Halappanavar, Joseph B. Manzano, and Antonino Tumeo. Optimizing irregular applications for energy and performance on the tilera many-core architecture. In *Proceedings of the ACM International Conference on Computing Frontiers*, Ischia, Italy, May 2015.

Daniel Chavarria-Miranda, Mahantesh Halappanavar, and Ananth Kalyanaraman. Scaling graph community detection on the tilera many-core architecture. In *The 21st International Conference on High Performance Computing*, pages 1–11. IEEE, 2014.

Linchuan Chen, Xin Huo, Bin Ren, Surabhi Jain, and Gagan Agrawal. Efficient and simplified parallel graph processing over CPU and MIC. In *The IEEE International Parallel and Distributed Processing Symposium*, pages 819–828. IEEE, 2015.

Aaron Clauset, Mark. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, December 2004.

Richard C. Connor, Michael R. Heithaus, and Lynne M. Barre. Superalliance of bottlenose dolphins. *Nature*, 397(6720):571–572, 1999.

Jeff Daily, Ananth Kalyanaraman, Sriram Krishnamoorthy, and Abhinav Vishnu. A work stealing based approach for enabling scalable optimal sequence homology detection. *Journal of Parallel and Distributed Computing*, 79:132–142, 2015.

Timothy A. Davis and Yifan Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, 38(1):1:1–1:25, 2011.

Sujay Deb, Amlan Ganguly, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. Wireless NoC as interconnection backbone for multicore chips: Promises and challenges. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(2):228–239, 2012.

Sujay Deb, Kuo-Pin Chang, Xinmin Yu, Suman P. Sah, Marijo Cosic, Anshuman Ganguly, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. Design of an energy-efficient CMOS-compatible NoC architecture with millimeter-wave wireless interconnects. *IEEE Transactions on Computers*, 62(12):2382–2396, 2013.

Karthi Duraisamy, Ryan Gary Kim, and Partha Pratim Pande. Enhancing performance of wireless NoCs with distributed mac protocols. In *The 16th International Symposium on Quality Electronic Design*, pages 406–411, 2015a.

Karthi Duraisamy, Hao Lu, Partha Pratim Pande, and Ananth Kalyanaraman. High Performance and Energy Efficient Wireless NoC-Enabled Multicore Architectures for Graph Analytics. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 147–156, 2015b.

Paul Erdös and Alfréd Rényi. {On the evolution of random graphs}. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5): 75–174, February 2010.

Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99 (12):7821–7826, 2002.

Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, pages 1360–1380, 1973.

Linley Gwennap. Adapteva: More flops, less watts. *Microprocessor Report*, 6 (13):11–02, 2011.

Pawan Harish and P. J. Narayanan. Accelerating large graph algorithms on the GPU using CUDA. In *High Performance Computing–HiPC 2007*, pages 197–208. Springer, 2007.

Kenneth A. Hawick, Arno Leist, and Daniel P. Playne. Parallel graph component labelling with GPUs and CUDA. *Parallel Computing*, 36(12):655–678, 2010.

Sungpack Hong, Tayo Oguntebi, and Kunle Olukotun. Efficient parallel graph exploration on multi-core CPU and GPU. In *The International Conference on Parallel Architectures and Compilation Techniques*, pages 78–88. IEEE, 2011.

Nicolai M. Josuttis. *The C++ Standard Library: A tutorial and reference*. Addison-Wesley Professional, 2012.

Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

George Karypis. METIS and ParMETIS. In *Encyclopedia of Parallel Computing*, pages 1117–1124. Springer, 2011.

George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, December 1998.

William O. Kermack and Anderson G. McKendrick. A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 115, pages 700–721. The Royal Society, 1927.

Sameer Kulkarni and John Cavazos. Mitigating the compiler optimization phase-ordering problem using machine learning. *ACM SIGPLAN Notices*, 47(10):147–162, 2012.

Amit Kumar, Li-Shiuan Peh, Partha Kundu, and Niraj K. Jha. Toward ideal on-chip communication using express virtual channels. *IEEE Micro*, (1): 80–90, 2008.

Dominique LaSalle and George Karypis. Multi-threaded modularity based graph clustering using the multilevel paradigm. *Journal of Parallel and Distributed Computing*, 76:66–80, 2015.

Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *The 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–480, 2009.

Louvain. findcommunities. URL `https://sites.google.com/site/findcommunities/`.

Hao Lu, Mahantesh Halappanavar, Daniel Chavarria-Miranda, Assefaw Gebremedhin, and Ananth Kalyanaraman. Balanced coloring for parallel computing applications. In *The IEEE International Parallel and Distributed Processing Symposium*, pages 7–16. IEEE, 2015a.

Hao Lu, Mahantesh Halappanavar, and Ananth Kalyanaraman. Parallel heuristics for scalable community detection. *Parallel Computing*, 47:19–37, 2015b.

Lijuan Luo, Martin Wong, and Wen-mei Hwu. An effective GPU implementation of breadth-first search. In *Proceedings of the 47th Design Automation Conference*, pages 52–55. ACM, 2010.

Kamesh Madduri, David Ediger, Karl Jiang, David Bader, Daniel Chavarria-Miranda. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8. IEEE, 2009.

Fredrik Manne and Rob H. Bisseling. A parallel approximation algorithm for the weighted maximum matching problem. In *Parallel Processing and Applied Mathematics*, pages 708–717. Springer, 2008.

Radu Marculescu, Umit Y. Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(1):3–21, 2009.

Duane Merrill, Michael Garland, and Andrew Grimshaw. Scalable GPU graph traversal. In *ACM SIGPLAN Notices*, volume 47, pages 117–128. ACM, 2012.

Duane Merrill, Michael Garland, and Andrew Grimshaw. High-performance and scalable GPU graph traversal. *ACM Transactions on Parallel Computing*, 1(2):14, 2015.

Stanley Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

Richard C. Murphy, Kyle B. Wheeler, Brian W. Barrett, and James A. Ang. Introducing the graph 500. *Cray Users Group (CUG)*, 2010.

Mark. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70 (5):056131, November 2004a.

Mark. E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004b.

Umit Y. Ogras and Radu Marculescu. Application-specific network-on-chip architecture customization via long-range link insertion. In *The IEEE/ACM International Conference on Computer-Aided Design*, pages 246–253, 2005.

Umit Y. Ogras and Radu Marculescu. "It's a small world after all": NoC performance optimization via long-range link insertion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(7):693–706, 2006.

Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012.

P. Krishna Reddy, Masaru Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *Databases in Networked Information Systems*, pages 188–200. Springer, 2002.

E. Jason Riedy, Henning Meyerhenke, David Ediger, and David A. Bader. Parallel community detection for massive graphs. In *Parallel Processing and Applied Mathematics*, pages 286–296. Springer, 2012. URL `http://link.springer.com/chapter/10.1007/978-3-642-31464-3_29`.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Ray Solomonoff and Anatol Rapoport. Connectivity of random nets. *The Bulletin of Mathematical Biophysics*, 13(2):107–117, 1951.

Jyothish Soman and Ankur Narang. Fast community detection algorithm with GPUs and multicore architectures. In *Parallel and Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 568–579. IEEE, 2011.

Jyothish Soman, Kothapalli Kishore, and P. J. Narayanan. A fast GPU algorithm for graph connectivity. In *2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8, 2010.

Christian L. Staudt and Henning Meyerhenke. Engineering high-performance community detection heuristics for massive graphs. In *The 42nd International Conference on Parallel Processing*, pages 180–189. IEEE, 2013.

Karsten Steinhaeuser, Nitesh V. Chawla, and Auroop R. Ganguly. Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):497–511, 2011.

Vincent A. Traag, Paul Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1): 016114, 2011.

Duncan J. Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press, 1999.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

Paul Wettin, Ryan Kim, Jacob Murray, Xinmin Yu, Partha Pratim Pande, Anshuman Ganguly, and Deukhyoun Heoamlan. Design space exploration for wireless nocs incorporating irregular network routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33 (11):1732–1745, 2014.

Udny G. Yule. A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, FRS. *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character*, pages 21–87, 1925.

Jianlong Zhong and Bingsheng He. Medusa: Simplified graph processing on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 25(6): 1543–1552, 2014.