Report from Dagstuhl Seminar 14461

High-performance Graph Algorithms and Applications in Computational Science

Edited by

Ulrich Carsten Meyer[1], Henning Meyerhenke[2], Ali Pinar[3], and Ilya Safro[4]

1 **Goethe-Universität Frankfurt am Main, DE,** umeyer@cs.uni-frankfurt.de
2 **Karlsruhe Institute of Technology (KIT), DE,** meyerhenke@kit.edu
3 **Sandia National Laboratories – Livermore, US,** apinar@sandia.gov
4 **Clemson University, US,** isafro@clemson.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 14461 "High- performance Graph Algorithms and Applications in Computational Science". The seminar reflected the recent qualitative change how graph algorithms are used in practice due to (i) the complex structure of graphs in new and emerging applications, (ii) the size of typical inputs, and (iii) the computer systems on which graph problems are solved. This change is having a tremendous impact on the field of graph algorithms in terms of algorithm theory and implementation as well as hardware requirements and application areas.

The seminar covered recent advances in all these aspects with a focus on practical algorithms and their efficient implementation for large-scale problems. The abstracts included in this report contain recent state-of-the-art results, but also point to promising new directions for high-performance graph algorithms and their applications.

Seminar November 10–14, 2014 – http://www.dagstuhl.de/14461
1998 ACM Subject Classification F.2.2 Graph Theory, G.2.2 Graph algorithms, D.1.3 Concurrent Programming, E.1 Data Structures – Graphs and networks, I.1.2 Algorithms – Algebraic algorithms
Keywords and phrases graphs, graph algorithms, graph theory, computational science, complex networks, network science, graph partitioning, linear algebra, parallel programming
Digital Object Identifier 10.4230/DagRep.4.11.40
Edited in cooperation with Christian L. Staudt

1 Executive Summary

Ulrich Carsten Meyer
Henning Meyerhenke
Ali Pinar
Ilya Safro

Many presentations in this Dagstuhl seminar emphasized recent trends regarding typical inputs and their effect on graph algorithm development for practical purposes. One can divide these presentations into four categories: (i) Traditional graph (or matrix) problems in new scenarios, (ii) graph analytics algorithms of various sorts, (iii) parallel computing aspects

High-performance Graph Algorithms and Applications in Computational Science, *Dagstuhl Reports*, Vol. 4, Issue 11, pp. 40–58
Editors: Ulrich Carsten Meyer, Henning Meyerhenke, Ali Pinar, and Ilya Safro

Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

such as tools, computational models, load balancing, and communication; and finally (iv) emerging high-performance application and hardware trends. The following four paragraphs give a brief overview over the talks presented in each of these categories.

Pothen discussed different matching problems and how the emergence of complex networks have changed various matching algorithms recently. Road networks, in turn, are by no means complex and the traditional Dijkstra algorithm solves queries on continental instances in few seconds. Yet, for more challenging scenarios, for example millions of queries per second on webservers or multiple optimization critieria, more elaborate solutions are necessary, as presented by Sanders. Toledo addressed the importance of communication efficiency on large-scale parallel systems for traditional numerical problems such as LU decomposition. A similar numerical topic was the solution of Laplacian linear systems, for which new combinatorial solvers and related techniques from the theory community were presented and discussed by Madry and by Toledo. Furthermore, Boman and Toledo initiated a tangible plan for a scientific competition on solvers for this class of linear systems.

The analytics algorithms part experienced a number of talks on graph clustering and community detection, which means the identification of natural vertex groups in graphs. Several very fast algorithms and their implementation were discussed and compared. Centralities are used for finding important (but in general unrelated) vertices or edges in a graph. Çatalyürek showed how to exploit parallelism in centrality algorithms to speed them up in different hardware settings, including accelerators. Bergamini, in turn, used approximation to obtain a speedup in dynamic graphs. Many other analytics tasks and algorithms were discussed, including anomaly detection presented by Miller and label inference by Chakrabarti, who both focused on techniques for very large graphs. Graph size was also a motivation for sparsification as discussed by Parthasarathy, either to save space or running time (or both) in later stages of an algorithmic pipeline.

Parallelism was the common theme in the third category. Here we summarize algorithmic techniques such as load balancing by graph partitioning, computational models as well as tools and middleware. Several speakers outlined challenges and/or algorithmic solutions in graph partitioning, in particular for complex networks or massively parallel systems. It became also clear that the development of graph algorithms for massive inputs benefits from suitable computational models. An example is the parallel external memory model for which Meyer as well as Veith showed algorithmic solutions. Another prerequisite for efficient graph algorithms in practice is tool support, including building block standards (proposed by Buluc) and communication middleware (presented by Lumsdaine). The pros and cons of different tools were discussed in an animated manner with the co-located Dagstuhl seminar 14462 “Systems and Algorithms for Large-scale Graph Analytics” within a joint session. The organizers are confident that this discussion has led to a better understanding of each other’s community and their contributions. We also hope and think that this exchange will lead to an accelerated dissemination of the respective leading research results across community borders.

Finally, Brugger presented innovative hardware specifically designed to support certain graph algorithms. Talks with a particular focus on innovative applications from outside the core of computer science were presented by several speakers as well. Both Srivastav and Buluc, for example, described algorithms for sequence assembly, a problem in bioinformatics with massive data sets.

2 Table of Contents

3 Overview of Talks

3.1 Community Finding Graph Algorithms on Multicores

Deepak Ajwani (Bell Labs – Dublin, IE)

Joint work of Ajwani, Deepak; Duriakova, Erika; Hurley, Neil; Sala, Alessandra
Main reference E. Duriakova, N. Hurley, D. Ajwani, A. Sala, "Analysis of the semi-synchronous approach to large-scale parallel community finding," in Proc. of the 2nd ACM Conf. on Online Social Networks (COSN'14), pp. 51–62, 2014.
URL http://dx.doi.org/10.1145/2660460.2660474
URL http://cosn.acm.org/2014/files/cosn049f-duriakovaAemb.pdf

Community-finding in graphs is the process of identifying highly cohesive vertex subsets. Many community-finding algorithms are based on the optimisation of an objective through a process of *iterative local update* (ILU), in which vertices are successively moved to the community of one of their neighbours in order to achieve the highest local gain in the quality of the objective. The sequential processing of such iterative algorithms generally benefits from an asynchronous approach, where a vertex update uses the most recent state as generated by the previous update of vertices in its neighbourhood. When vertices are distributed over a parallel machine, the asynchronous approach can encounter race conditions that impact on its performance and destroy the consistency of the results. Alternatively, a semi-synchronous approach ensures that only non-conflicting vertices are updated simultaneously. In this talk, I present our work on the semi-synchronous approach to ILU algorithms for community finding on social networks. Because of the heavy-tailed vertex distribution, the order in which vertex updates are applied in asynchronous ILU can greatly impact on both convergence time and quality of the found communities. We study the impact of ordering on the distributed label propagation and modularity maximisation algorithms implemented on a shared- memory multicore architecture. We demonstrate that the semi-synchronous ILU approach is competitive in time and quality with the asynchronous approach, while allowing the analyst to maintain consistent control over update ordering. Thus, our implementation results in a more robust and predictable performance and provides control over the order in which the node labels are updated, which is crucial to obtaining the correct trade-off between running time and quality of communities on many graph classes.

3.2 Approximating Betweenness Centrality in Large Evolving Networks

Elisabetta Bergamini (KIT – Karlsruher Institut für Technologie, DE)

Joint work of Bergamini, Elisabetta; Meyerhenke, Henning; Staudt, Christian L.
Main reference E. Bergamini, H. Meyerhenke, C. L. Staudt, "Approximating Betweenness Centrality in Large Evolving Networks," in Proc. of the 17th Workshop on Algorithm Engineering and Experiments (ALENEX'15), pp. 133–146, SIAM, 2015; pre-print available as arXiv:1409.6241v1 [cs.SI].
URL http://dx.doi.org/10.1137/1.9781611973754.12
URL http://arxiv.org/abs/1409.6241v1

Betweenness centrality ranks the importance of nodes by their participation in all shortest paths of the network. Therefore computing exact betweenness values is impractical in large networks. For static networks, approximation based on randomly sampled paths has been shown to be significantly faster in practice. However, for dynamic networks, no approximation algorithm for betweenness centrality is known that improves on static recomputation. We

address this deficit by proposing two incremental approximation algorithms (for weighted and unweighted connected graphs) which provide a provable guarantee on the absolute approximation error. Processing batches of edge insertions, our algorithms yield significant speedups up to a factor of 104 compared to restarting the approximation. This is enabled by investing memory to store and efficiently update shortest paths. As a building block, we also propose an asymptotically faster algorithm for updating the SSSP problem in unweighted graphs. Our experimental study shows that our algorithms are the first to make in-memory computation of a betweenness ranking practical for million-edge semi-dynamic networks. Moreover, our results show that the accuracy is even better than the theoretical guarantees in terms of absolutes errors and the rank of nodes is well preserved, in particular for those with high betweenness.

3.3 2D Partitioning for Scalable Matrix Computations on Scale-Free Graphs

Erik Boman (Sandia National Laboratories – Albuquerque, US)

Joint work of Boman, Erik; Devine, Karen; Rajamanickam, Sivasankaran
Main reference E. G. Boman, K. D. Devine, S. Rajamanickam, "Scalable Matrix Computations on Scale-Free Graphs Using 2D Graph Partitioning," in Proc. of the 2013 Int'l Conf. on High Performance Computing, Networking, Storage and Analysis (SC'13), Article No. 50, 12 pages, 2013.
URL http://dx.doi.org/10.1145/2503210.2503293

Scalable parallel computing is essential for processing large scale-free (power-law) graphs. The data distribution becomes important on distributed-memory computers with thousands of cores. Recently, it has been shown that 2D layouts (edge partitions) have significant advantages over traditional 1D layouts. However, the simple 2D block distribution does not use the structure of the graph, and more advanced 2D partitioning methods are too expensive for large graphs. We propose a new partitioning algorithm that combines graph or hypergraph partitioning with the 2D block distribution. The cost is essentially the same as 1D (hyper-)graph partitioning. We study the performance of sparse matrix-vector multiplication for large scale-free graphs from, e. g., social networks using several partitioners and data layouts, both 1D and 2D. We demonstrate that our new 2D method consistently outperforms the other methods considered, both for SpMV and an eigensolver, on matrices up to 1.6 billion non-zeros and up to 16,384 cores. We leave as future work a comparison to other 2D matrix partitioning methods that are not available in parallel software. (This work was first presented at SC13.)

3.4 Beyond the abstract machine model – How looking at real computing systems leads to new algorithmic insights and massive speedups: two case studies

Christian Brugger (TU Kaiserslautern, DE)

Abstract machine models have a simplistic view on computing systems, often assuming all operations having the same cost. While this is useful for asymptotic complexity analysis, they can be misleading when trying to find the best algorithms for finite datasets. In fact

it is not uncommon to have a 3 orders of magnitude difference in the cost of two similar sounding operations. In this talk we will present a more realistic view of today's computing systems. Cover communication costs, operator costs, memory models and data formats. Based on these insights we will look at two case studies. In them we show how it is possible to formulate new algorithms exploiting these differences, resulting in faster implementations.

3.5 The Graph BLAS: building blocks for graph algorithms in the language of linear algebra

Aydin Buluc (Lawrence Berkeley National Laboratory, US)

Joint work of Buluc, Aydin; Bader, David; Gilbert, John; Gonzalez, Joseph; Kepner, Jeremy; Mattson, Timothy
Main reference T. Mattson, D. Bader, J. Berry, A. Buluc, J. Dongarra, C. Faloutsos, J. Feo, J. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. Leiserson, A. Lumsdaine, D. Padua, S. Poole, S. Reinhardt, M. Stonebraker, S. Wallach, A. Yoo, "Standards for graph algorithm primitives," in Proc. of the 2013 IEEE High Performance Extreme Computing Conference (HPEC'13), pp. 1–2, IEEE, 2013.
URL http://dx.doi.org/10.1109/HPEC.2013.6670338

We believe that the state of the art in constructing a large collection of graph algorithms in terms of linear algebraic operations is mature enough to support the emergence of a standard set of primitive building blocks. It is critical to move quickly and define such a standard; thereby freeing up researchers to innovate and diversify at the level of higher level algorithms and graph analytics applications. This effort was inspired by the Basic Linear Algebra Subprograms (BLAS) of dense linear algebra and hence our working name for this standard is "the Graph BLAS". This talk will cover the rationale, minimal requirements, existing tools, best practices, and wish lists.

3.6 Fast Graph Centrality Computations

Ümit V. Çatalyürek (Ohio State University, US)

Joint work of Sariyuce, Ahmet E.; Saule, Erik; Kaya, Kamer; Çatalyürek, Ümit V.
Main reference A. E. Sariyüce, E. Saule, K. Kaya, Ü. V. Çatalyürek, "Regularizing Graph Centrality Computations," Journal of Parallel and Distributed Computing, 2014.
URL http://dx.doi.org/10.1016/j.jpdc.2014.07.006

Centrality metrics such as betweenness and closeness have been used to identify important nodes in a network. However, it takes days to months on a high-end workstation to compute the centrality of today's networks. The main reasons are the size and the irregular structure of these networks. While today's computing units excel at processing dense and regular data, their performance is questionable when the data is sparse. In this talk, we show how centrality computations can be regularized to reach higher performance.

3.7 Joint Inference of Multiple Label Types in Large Networks

Deepayan Chakrabarti (Facebook – Menlo Park, US)

Joint work of Chakrabarti, Deepayan; Funiak, Stanislav; Chang, Jonathan; Macskassy, Sofus
Main reference D. Chakrabarti, S. Funiak, J. Chang, S. A. Macskassy, "Joint Inference of Multiple Label Types in Large Networks," in Proc. of the 31th Int'l Conf. on Machine Learning (ICML'14), JMLR Proceedings, Vol. 32, pp. 874–882, 2014.
URL http://jmlr.org/proceedings/papers/v32/chakrabarti14.html

We tackle the problem of inferring node labels in a partially labeled graph where each node in the graph has multiple label types and each label type has a large number of possible labels. Our primary example, and the focus of this paper, is the joint inference of label types such as hometown, current city, and employers, for users connected by a social network. Standard label propagation fails to consider the properties of the label types and the interactions between them. Our proposed method, called EdgeExplain, explicitly models these, while still enabling scalable inference under a distributed message-passing architecture. On a billion-node subset of the Facebook social network, EdgeExplain significantly outperforms label propagation for several label types, with lifts of up to 120% for recall@1 and 60% for recall@3.

3.8 GEMS – a scalable triplestore for unstructured, heterogeneous data sets

John Feo (Pacific Northwest National Lab. – Richland, US)

Data collection and analysis are rapidly changing the way scientific, national security, and business communities operate. Data is no longer "owner generated," but rather collected from web sources. American economic competitiveness and security depend increasingly on the insightful analysis of unstructured, heterogeneous web-scale data sets. The fixed schemas and tables of relational database do not support unstructured data. NoSQL database do a better job, but are poor at processing joins operations. Neither type of database naturally supports subgraph isomorphism, typed path traversal, and community detection. To perform such complex graph analytics, analysts export a small snapshot of their data into a single system image restricting their global view and paying a steep price in operational requirements. Even so, many analytical capabilities, such as determining behavior from structure, lie out of reach due to the lack of computational power.

In response to these analytic challenges, we are developing GEMS, a scalable triplestore for unstructured, heterogeneous data. The systems has three components: 1) a SPARQL front end to transform SPARQL to data parallel C code; 2) a semantic graph engine with scalable multithreaded algorithms for query processing; and 3) a custom multithreaded runtime layer for scalable performance on conventional cluster systems. Our objectives are twofold: 1) to scale system size as data sizes increase, and 2) to maintain query throughput as system size grows. We are accomplishing these objectives by targeting conventional clusters with large memory nodes, developing an in-memory graph engine, managing a fine-grain multithreaded runtime layer to hide memory latencies, and aggressively aggregating memory requests to maximize system bandwidth.

In this talk, I will discuss the data challenges facing scientists, intelligence analysts, and business leaders. I will describe the GEMS architecture focusing on the graph engine and runtime layer. I will present query patterns in cyber security, fraud, and supply chains, and performance results comparing GEMS to commercial systems.

3.9 Multi-Threaded Modularity Based Graph Clustering using the Multilevel Paradigm

Dominique LaSalle (University of Minnesota – Minneapolis, US)

Joint work of LaSalle, Dominique; Karypis, George
Main reference D. LaSalle, G. Karypis, "Multi-threaded modularity based graph clustering using the multilevel paradigm," Journal of Parallel and Distributed Computing, 2014.
URL http://dx.doi.org/10.1016/j.jpdc.2014.09.012

Graphs are an important tool for modeling data in many diverse domains. Recent increases in sensor technology and deployment, the adoption of online services, and the scale of VLSI circuits has caused the size of these graphs to skyrocket. Finding clusters of highly connected vertices within these graphs is a critical part of their analysis. In this work we apply the multilevel paradigm to the modularity graph clustering problem. We present fast shared-memory parallel algorithms for modularity maximization that produce clusterings of high quality. The implementation of these algorithms, Nerstrand, runs in a fraction of the time of current methods and exhibits significant speedup with less than one percent degradation of clustering quality of its serial counterpart. Nerstrand works well on large graphs, clustering a graph with over 105 million vertices and 3.3 billion edges in 90 seconds.

3.10 PULP: Fast and Simple Complex Network Partitioning

Kamesh Madduri (Pennsylvania State University – University Park, US)

Joint work of Slota, George; Madduri, Kamesh; Rajamanickam, Sivasankaran
Main reference G. M. Slota, K. Madduri, S. Rajamanickam, "PuLP: Scalable multi-objective multi-constraint partitioning for small-world networks," in Proc. of the 2014 IEEE Int'l Conf. on Big Data (BigData'14), pp. 481–490, IEEE, 2014.
URL http://dx.doi.org/10.1109/BigData.2014.7004265

Complex networks such as web crawls and social networks are known to lack good separators. The common practice in the community is to use well-known graph and hypergraph partitioners as black-box routines and hope that these tools produce good partitions (i. e., low edge cut or communication volume, while maintaining vertex balance). We argue that it is unnecessary to use existing multilevel tools for partitioning complex networks, as these tend to be quite memory- and compute-intensive for graphs with billions of vertices and edges. We design a simple and easy-to-configure parallel graph partitioner called PULP (Partitioning using Label Propagation). As the name suggests, PULP uses a "label propagation"-like initial partitioning strategy. Label propagation is a popular heuristic for the community detection problem. PULP then uses the Fiduccia-Mattheyses heuristic for refining the partitions. PULP simultaneously optimizes for multiple quality measures (total edge cut and max per-partition edge cut), while satisfying user-defined balance constraints on per-partition

edge and vertex counts. It's extremely fast: for instance, on the 1.8 billion edge Slovakian domain web crawl, PULP takes less than a minute on a single compute node, to generate a 128-way vertex partitioning. For partitioning web crawls, we find that the quality of results is comparable to existing tools.

3.11 Cuts, Trees, and Electrical Flows

Aleksander Madry (EPFL – Lausanne, CH)

Main reference A. Madry, "Fast Approximation Algorithms for Cut-based Problems in Undirected Graphs," in Proc. of the 51st Annual IEEE Symp. on Foundations of Computer Science (FOCS'10), pp. 245–254, IEEE CS, 2010; pre-print available as arXiv:1008.1975v4 [cs.DS].
URL http://dx.doi.org/10.1109/FOCS.2010.30
URL http://arxiv.org/abs/1008.1975v4

We discuss some of the recent developments in algorithmic graph theory that might be relevant in the context of dealing with massive graphs.

In particular, we present a general framework for obtaining close-to-linear-time approximation algorithms for cut problems in undirected graph. We also discuss the electrical flow paradigm that played key role in some of the recent progress on designing fast algorithms for fundamental flow problems.

3.12 Fast generation of dynamic complex networks with underlying hyperbolic geometry

Henning Meyerhenke (KIT – Karlsruher Institut für Technologie, DE)

Joint work of von Looz, Moritz; Meyerhenke, Henning; Staudt, Christian L.; Prutkin, Roman
Main reference M. von Looz, C. L. Staudt, H. Meyerhenke, R. Prutkin, "Fast generation of dynamic complex networks with underlying hyperbolic geometry," Karlsruhe Reports in Informatics; 2014,14.
URL http://digbib.ubka.uni-karlsruhe.de/volltexte/1000043881

The analysis of *complex networks* has become a highly active research area recently since complex networks are increasingly used to represent phenomena as varied as the WWW, social relations, protein interactions, and brain topology. Complex networks are usually *scale-free*, their degree distribution often follows a power law, and the typical distance between two nodes is surprisingly small, regardless of network size and growth. *Generative network models* play a central role in many complex network studies for several reasons: Real data, such as social networks, might contain confidential information, so that it is often desirable to be able to work on similar synthetic networks. Quick testing of algorithms requires small test cases, while projection of future growth and scalability studies need bigger graphs. Moreover, real networks might be impractical to transmit and store, but compression to the parameters of a model is possible.

One generative network model that has been suggested previously as fairly realistic (Krioukov et al. 2010) creates unit-disk graphs in *hyperbolic geometry*. Among the many interesting properties of hyperbolic geometry, most relevant is the *exponential expansion of space*: The area of a hyperbolic circle of radius r is $2\pi(\cosh(r) - 1) \simeq e^r$, allowing a natural embedding of trees and tree-like graphs. In recent years, the link between hyperbolic

geometry and graphs with power-law degree distributions has been studied with respect to routing applications (Boguña et al. 2010). The generative model has a proven high clustering coefficient (Gugelmann et al. 2012), small diameter and a power-law degree distribution with adjustable exponent (Krioukov et al. 2010).

The model distributes nodes randomly on a hyperbolic disk of radius R and edges are inserted for every node pair whose hyperbolic distance is below a threshold. Calculating the hyperbolic distance between each pair of coordinates has quadratic time complexity. This impedes the creation of massive networks and is likely the reason previously published networks based on hyperbolic geometry have been in the range of at most 10^4 nodes. A faster generator is necessary to enable a use of this promising model for networks of interesting scales. Additionally, to judge the realism of these networks, more detailed parameter studies and comparisons from a network analysis point of view are necessary.

As part of our study, we address deficiencies of the *hyperbolic unit-disk graph model* in terms of generation speed and network analysis. First we show how we relate hyperbolic to Euclidean geometry during the generation process. This allows us to employ a new space-partitioning data structure, more precisely a polar quadtree within the Poincaré disk model, to improve the running time of the naive generation process. We proceed by proposing an alternative dynamic model. Instead of deleting and reinserting nodes (Papadopoulos 2010), we let nodes move gradually in the hyperbolic plane. This results in a smoother change of the network, so that we believe it to be more realistic for some applications. We also analyze the time complexity of our static and dynamic generation process, resulting in an expected static running time in $O((n+m)\log n)$ and an expected dynamic running time in $O((k+l)\log n)$ when moving k nodes with l edges under a reasonable assumption.

Finally, we add to previous studies a comprehensive network analytic evaluation of the generative model based on hyperbolic geometry. The experimental results confirm the theoretical expected running time of $O((n+m)\log n)$. In practice, a graph with 10^7 nodes and 10^9 edges can be generated in less than 5 minutes on our test machine. Network analysis shows a consistently high clustering coefficient and power-law degree distribution over a wide parameter range. The generator will be made available in a future version of NetworKit (Staudt et al. 2013), our open-source framework for large-scale network analysis.

3.13 External memory graph algorithms

Ulrich Carsten Meyer (Goethe-Universität Frankfurt am Main, DE)

Large graphs arise naturally in many real world applications. The actual performance of simple RAM model algorithms for traversing these graphs (stored in external memory) deviates significantly from their linear or near-linear predicted performance because of the large number of I/Os they incur. In order to alleviate the I/O bottleneck, many external memory graph traversal algorithms have been designed with provable worst-case guarantees. In the talk I highlight some techniques used in the design and engineering of such algorithms and survey the state-of-the-art in I/O-efficient graph traversal algorithms.

3.14 Spectral Anomaly Detection in Very Large Graphs: Models, Noise, and Computational Complexity

Benjamin A. Miller (MIT Lincoln Laboratory – Lexington, US)

Joint work of Miller, Benjamin A.; Arcolano, Nicholas; Wolf, Michael M.; Bliss, Nadya T.
Main reference B. A. Miller, N. Arcolano, M. M. Wolf, N. T. Bliss, "Spectral Anomaly Detection in Very Large Graphs: Models, Noise, and Computational Complexity," arXiv:1412.4411v1 [cs.SI], 2014.
URL http://arxiv.org/abs/1412.4411v1

Anomaly detection in massive networks has numerous theoretical and computational challenges, especially as the behavior to be detected becomes small in comparison to the larger network. This presentation focuses on recent results in three key technical areas, specifically geared toward spectral methods for detection. We first discuss recent models for network behavior, and how their structure can be exploited for efficient computation of the principal eigenspace of the graph [1]. In addition to the stochasticity of background activity, a graph of interest may be observed through a noisy or imperfect mechanism, which may hinder the detection process. A few simple noise models are discussed, and we demonstrate the ability to fuse multiple corrupted observations and recover detection performance [2]. Finally, we discuss the challenges in scaling the spectral algorithms to large-scale high-performance computing systems, and present preliminary recommendations to achieve good performance with current parallel eigensolvers [3].

References

1 Benjamin A. Miller, Nicholas Arcolano, and Nadya T. Bliss. Efficient anomaly detection in dynamic, attributed graphs. In *Proc. IEEE Intelligence and Security Informatics*, pages 179–184, 2013.
2 B. A. Miller and N. Arcolano. Spectral subgraph detection with corrupt observations. In *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, pages 3449–3453, 2014.
3 M. M. Wolf and B. A. Miller. Sparse matrix partitioning for parallel eigenanalysis of large static and dynamic graphs. In *Proc. IEEE High Performance Extreme Computing Conf.*, 2014.

3.15 Practical Graph Sparsification

Srinivasan Parthasarathy (Ohio State University – Columbus, US)

Joint work of Parthasarathy, Srinivasan; Satuluri, Venu; Ruan, Yiye; Fuhry, Dave; Zhang Yang
Main reference V. Satuluri, S. Parthasarathy, Y. Ruan, "Local graph sparsification for scalable clustering," in Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data, pp. 721–732, ACM, 2011.
URL http://dx.doi.org/10.1145/1989323.1989399
Main reference Y. Ruan, D. Fuhry, S. Parthasarathy, "Efficient community detection in large networks using content and links," in Proc. of the 22nd Int'l World Wide Web Conf. (WWW'13), pp. 1089–1098, 2013.
URL http://dl.acm.org/citation.cfm?id=2488483

Many real world problems (biological, social, web) can be effectively modeled as networks or graphs where nodes represent entities of interest and edges mimic the interactions or relationships among them. The study of such complex relationship networks, recently referred to as "network science", can provide insight into their structure, properties and emergent behavior. Of particular interest here are rigorous methods for uncovering and understanding

important network structures and motifs (communities) at multiple topological and temporal scales. Achieving this objective is challenging due to the presence of noise (false or missing interactions), topological(scale-free)) properties and scalability. Given the importance of the graph clustering problem, a number of solutions ranging from hierarchical methods to spectral methods have been designed and developed.

In this talk I will discuss a novel approach to sparsifying or sampling the edges of a graph while retaining the relevant content and structure important to a range of graph processing or graph analytic tasks. Empirical results demonstrate both qualitative as well as quantitive improvements over existing approaches on a wide range of datasets drawn from socio- technological- and biological- domains. Time permitting, I will also illustrate the value of such an approach from the perspective of visually teasing out relevant structure from large scale graphs and networks.

3.16 Current challenges for parallel graph partitioning

Francois Pellegrini (University of Bordeaux, FR)

Graph partitioning is a technique used for the solving of many problems in scientific computing, such as the decomposition of a mesh into domains so as to evenly balance the compute load on the processors of a parallel architecture. Because of the ever increasing size of the meshes to handle, partitioning tools themselves had to be parallelized. The parallel versions of these software provide good results for and on several thousands of processors, but the advent of architectures comprising more than a million processing elements raises new problems. Not only do the partitioning results produced by these software have to take into account the heterogeneity of these architectures, but also does the efficient execution of the partitioning software on these architectures require much more sophisticated algorithms. The purpose of this talk is to present the challenges to overcome in order to reach these goals.

3.17 Sampling and streaming algorithms for counting small patterns in BIG graphs

Ali Pinar (Sandia National Laboratories – Livermore, US)

Joint work of Pinar, Ali; Seshadhri, C.; Jha, Madhav; Kolda, Tamara

Counting the frequency of small subgraphs is a fundamental technique in network analysis across various domains, most notably in bioinformatics and social networks. Computing these counts can be challenging due to the sizes of the graphs. We have designed sampling algorithms that can provide provably accurate estimates for counting 3- and 4-vertex patterns. These algorithms have error/confidence bounds that depend on the number of samples but independent of the sizes of the graphs. We also designed a sublinear memory streaming algorithm to count triangles. This talk will summarize our results.

3.18 Multiscale methods for networks

Ilya Safro (Clemson University, US)

Joint work of Achi Brandt; Jie Chen; Alexander Gutfraind; William Hager; James Hungerford; Sven Leyffer; Lauren Meyers; Dorit Ron; Ilya Safro; Boris Temkin
URL http://www.cs.clemson.edu/~isafro

The main objective of a multilevel algorithm is to create a hierarchy of problems, each representing the original problem, but with fewer degrees of freedom. We will discuss several recently developed scalable algorithms for network modeling, fast response to epidemics on networks, the minimum vertex separator, and the minimum logarithmic arrangement problems.

1. We introduce a flexible method for synthesizing realistic ensembles of networks starting from a known network, through a series of mappings that coarsen and later refine the network structure by randomized editing. The method, MUSKETEER, preserves structural properties with minimal bias, including unknown or unspecified features, while introducing realistic variability at multiple scales. Using examples from several domains, we show that MUSKETEER produces the intended stochasticity while achieving greater fidelity across a suite of network properties than do other commonly used network generation algorithms.
2. We present a strategy for designing fast and practical methods of response to cyber attacks and infection spread on complex weighted networks. In these networks, vertices can be interpreted as primitive elements of the system, and weighted edges reflect the strength of interaction among these elements. The proposed strategy belongs to the family of multiscale methods whose goal is to approximate the system at multiple scales of coarseness and to obtain a solution of microscopic scale by combining the information from coarse scales. We consider an optimization problem that is based on the susceptible-infected-susceptible (SIS) epidemiological model. The objective is to detect the network vertices that have to be secured (or immunized) in order to keep a low level of infection in the system.
3. The Vertex Separator Problem for a graph is to find the smallest collection of vertices whose removal breaks the graph into two disconnected subsets that satisfy specified size constraints. This problem can be formulated as a continuous (non-concave/non-convex) bilinear quadratic program. We develop a more general continuous bilinear program which incorporates vertex weights, and which applies to the coarse graphs that are generated in a multilevel compression of the original Vertex Separator Problem. A Mountain Climbing Algorithm is used to find a stationary point of the continuous bilinear quadratic program, while second-order optimality conditions and perturbation techniques are used to escape from either a stationary point or a local maximizer. Computational results and comparisons demonstrate the advantage of the proposed algorithm.
4. We present a fast multiscale approach for the network minimum logarithmic arrangement problem. This type of arrangement plays an important role in the network compression and fast node/link access operations. The algorithm is of linear complexity and exhibits good scalability, which makes it practical and attractive for use in large-scale instances. Its effectiveness is demonstrated on a large set of real-life networks. These networks with corresponding best- known minimization results are suggested as an open benchmark for the research community to evaluate new methods for this problem.

3.19 Multicriteria Shortest Paths

Peter Sanders (KIT – Karlsruher Institut für Technologie, DE)

Joint work of Sanders, Peter; Erb, Stephan, Mandow, Lawrence; Kobitzsch, Moritz
Main reference P. Sanders, L. Mandow, "Parallel Label-Setting Multi-Objective Shortest Path Search," in Proc. of the 2013 IEEE 27th Int'l Parallel & Distributed Processing Symposium, pp. 215–224, IEEE, 2013.
URL http://dx.doi.org/10.1109/IPDPS.2013.89

Tn this talk, we present a parallel algorithm for finding all Pareto optimal paths from a specified source in a graph. The algorithm is label-setting, i. e., it only performs work on distance labels that are optimal. The main result is that the added complexity when going from one to multiple objectives is completely parallelizable. The algorithm is based on a multi-objective generalization of a priority queue. Such a Pareto queue can be efficiently implemented for two dimen- sions. Surprisingly, the parallel biobjective approach yields an algo- rithm performing asymptotically less work than the previous sequen- tial algorithms. Using a Pareto queue based on B-trees with parallel bulk updates, this also turns out to be practical. We also discuss generalizations for 3 objective functions and for single target search.
This is a summary of two conference papers [1, 3].

References

1 Peter Sanders and Lawrence Mandow. Parallel label-setting multi-objective shortest path search. In *IPDPS*, pages 215–224, Washington, 2013. IEEE Computer Society.
2 Peter Sanders and Lawrence Mandow. *Parallel Label-Setting Multi-Objective Shortest Path Search*. 27th IEEE Int'l Parallel & Distributed Processing Symp., Boston, USA, 2013.
3 Stephan Erb, Moritz Kobitzsch, and Peter Sanders. Parallel bi-objective shortest paths using weight-balanced b-trees with bulk updates. In *Symposium on Experimental Algorithms (SEA)*, Lecture Notes in Computer Science. Springer, 2014.

3.20 Parallel Graph Partitioning for Complex Networks

Christian Schulz (KIT – Karlsruher Institut für Technologie, DE)

Joint work of Meyerhenke, Henning; Sanders, Peter; Schulz, Christian
Main reference H. Meyerhenke, P. Sanders, C. Schulz, "Parallel Graph Partitioning for Complex Networks," arXiv:1404.4797v3 [cs.DC] , 2015.
URL http://arxiv.org/abs/1404.4797v3

Processing large complex networks like social networks or web graphs has recently attracted considerable interest. To do this in parallel, we need to partition them into pieces of about equal size. Unfortunately, previous parallel graph partitioners originally developed for more regular mesh-like networks do not work well for these networks. This talk addresses this problem by parallelizing and adapting the label propagation technique originally developed for graph clustering. By introducing size constraints, label propagation becomes applicable for both the coarsening and the refinement phase of multilevel graph partitioning. We obtain very high quality by applying a highly parallel evolutionary algorithm to the coarsest graph. The resulting system is both more scalable and achieves higher quality than state-of-the-art systems like ParMetis or PT-Scotch. For large complex networks the performance differences are very big. As an example, our algorithm partitions a web graph with 3.3G edges in 16

seconds using 512 cores of a high-performance cluster while producing a high quality partition – none of the competing systems can handle this graph on our system.

3.21 Tools for the Analysis of Large Networks: Algorithms and Software

Christian Staudt (KIT – Karlsruher Institut für Technologie, DE)

Joint work of Staudt, Christian; Meyerhenke, Henning; Sazonovs, Aleksejs; Bergamini, Elisabetta; von Looz, Moritz; Lindner, Gerd; Hamann, Michael
Main reference C. L. Staudt, A. Sazonovs, H. Meyerhenke, "NetworKit: An Interactive Tool Suite for High-Performance Network Analysis," arXiv:1403.3005v2 [cs.SI], 2014.
URL http://arxiv.org/abs/1403.3005v2

Network science can be defined as an interdisciplinary mathematical science studying the statistics, structure and dynamics of complex relational data. While complex network models are applicable to all kinds of domains, the commonality is the observation that the structure of relationships between entities allows important insights into complex systems. In practice, network scientists often need to perform exploratory analysis workflows on massive graph data sets. As computer scientists we can support this emerging field of research by providing appropriate computational tools, which includes both effective and efficient graph algorithms and usable data analysis software. This is what we are trying to do with NetworKit, an open-source tool suite for high-performance network analysis: Our goal is to package current results of our algorithm engineering efforts and put them into the hands of domain experts. NetworKit has a hybrid architecture with a C++/OpenMP backend and a Python frontend. Scaling to massive networks is enabled by methods such as parallel and sampling-based approximation algorithms. The current feature set includes various analytics algorithms, e. g. for community detection, and graph generators. Recent projects extending NetworKit include: 1) A collection of methods for sparsifying complex networks while preserving certain structural properties, a kind of lossy compression for massive networks; 2) A parallel generator for large synthetic networks based on the unit-disk graph model in hyperbolic geometry. In the future we would like to automate many network analysis steps to test for interesting correlations and uncover the significant features of a network. NetworKit is free software, open to a diverse community of algorithm engineers and data analysts.

3.22 Communication Efficient LU with Partial Pivoting using a Shape Morphing Data Layout

Sivan Toledo (Tel Aviv University, IL)

Joint work of Ballard, Grey; Demmel, James; Lipshitz, Benjamin; Schwartz, Oded; Toledo, Sivan
Main reference G. Ballard, J. Demmel, B. Lipshitz, O. Schwartz, S. Toledo, "Communication Efficient Gaussian Elimination with Partial Pivoting using a Shape Morphing Data Layout," in Proc. of the 25th Annual ACM Symp. on Parallelism in Algorithms and Architectures (SPAA'13), pp. 232–241, ACM, 2013.
URL http://dx.doi.org/10.1145/2486159.2486198

High performance for numerical linear algebra often comes at the expense of stability. Computing the LU decomposition of a matrix via Gaussian Elimination can be organized so that the computation involves regular and efficient data access. However, maintaining

numerical stability via partial pivoting involves row interchanges that lead to inefficient data access patterns. To optimize communication efficiency throughout the memory hierarchy we confront two seemingly contradictory requirements: partial pivoting is efficient with column-major layout, whereas a block-recursive layout is optimal for the rest of the computation. We resolve this by introducing a shape morphing procedure that dynamically matches the layout to the computation throughout the algorithm, and show that Gaussian Elimination with partial pivoting can be performed in a communication efficient and cache-oblivious way. Our technique extends to QR decomposition, where computing Householder vectors prefers a different data layout than the rest of the computation.

3.23 An I/O-efficient Distance Oracle for Evolving Real-World Graphs

David Veith (Goethe-Universität Frankfurt am Main, DE)

Joint work of Ajwani, Deepak; Meyer, Ulrich; Veith, David
Main reference D. Ajwani, U. Meyer, D. Veith, "An I/O-efficient Distance Oracle for Evolving Real-World Graphs," in Proc. of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX'15), pp. 159–172, SIAM, 2015.
URL http://dx.doi.org/10.1137/1.9781611973754.14

We present an I/O-efficient distance oracle that is able to answer online queries with a constant number of I/O. Furthermore, we developed batched queries that have an amortized I/O-complexity of $O(\frac{1}{B})$ I/Os per query. Online queries can be processed in milliseconds on SSDs and batched queries within microseconds even on HDDs. All results have been achieved on real world graphs. We explain the experimental results and discuss improvements for the future.

3.24 Assigning edge weights in graphs for quantofying vertex closeness

Panayot S. Vassilevski (LLNL – Livermore, US

Main reference Henson, Van Henson; Hysom, David; Sanders, Geoff; Vassilevski, Panayot; Yoo, Andy; "Assigning edge weights in graphs for quantifying vertex closeness," Lawrence Livermore National Laboratory Technical Report LLNL-JRNL-664198, November 13, 2014.

We propose an algorithm to assign edge-weights in graphs by minimizing a nonlinear functional. The functional is constructed in a way to expose if a vertex is closer to one of its neighbors than to the remaining ones. The functional is well-defined for both undirected and directed graphs, and is efficiently minimized for bipartite graphs. We also outline modifications of the functional applied to embeddings of the original graph into bipartite ones for which the minimization procedures are efficient (and parallelizable).

Based on the computed edge weights we design a recursive (multilevel) pairwise aggregation algorithm for community detection which breaks when a certain "energy" functional reaches a (local) minimum which reflects a balanced set of aggregates, in terms of edge weights. We also explore the use of the computed weights in a state-of-the-art multilevel aggregation-based community detection algorithm such as the Louvain algorithm that maximizes the popular graph modularity measure. The algorithms were applied to several graphs with known (full

or partial) ground-truth communities to verify the applicability of the computed edge weights to such more realistic situations.
The presentation is based on the results reported in [1].

References

1 V. E. Henson, D. Hysom, G. Sanders, P. S. Vassilevski, and A. Yoo, Assigning edge weights in graphs for quantifying vertex closeness. Lawrence Livermore National Laboratory Technical Report LLNL-JRNL-664198, November 13, 2014.

Participants

- Deepak Ajwani
Bell Labs – Dublin, IE
- Elisabetta Bergamini
KIT – Karlsruher Institut für Technologie, DE
- Rob Bisseling
Utrecht University, NL
- Erik Boman
Sandia National Laboratories – Albuquerque, US
- Christian Brugger
TU Kaiserslautern, DE
- Aydin Buluc
Lawrence Berkeley National Laboratory, US
- Ümit V. Çatalyürek
Ohio State University, US
- Deepayan Chakrabarti
Facebook – Menlo Park, US
- Tiago de Paula Peixoto
Universität Bremen, DE
- Yann Disser
TU Berlin, DE
- John Feo
Pacific Northwest National Lab. – Richland, US
- Enver Kayaaslan
ENS – Lyon, FR
- Dominique LaSalle
University of Minnesota – Minneapolis, US
- Andrew Lumsdaine
Indiana University – Bloomington, US
- Kamesh Madduri
Pennsylvania State University – University Park, US
- Aleksander Madry
EPFL – Lausanne, CH
- Fredrik Manne
University of Bergen, NO
- Ulrich Carsten Meyer
Goethe-Universität Frankfurt am Main, DE
- Friedhelm Meyer auf der Heide
Universität Paderborn, DE
- Henning Meyerhenke
KIT – Karlsruher Institut für Technologie, DE
- Benjamin A. Miller
MIT Lincoln Laboratory – Lexington, US
- Petra Mutzel
TU Dortmund, DE
- Braxton Osting
University of Utah, US
- Srinivasan Parthasarathy
Ohio State University – Columbus, US
- Francois Pellegrini
University of Bordeaux, FR
- Ali Pinar
Sandia National Laboratories – Livermore, US
- Alex Pothen
Purdue University, US
- Ilya Safro
Clemson University, US
- Peter Sanders
KIT – Karlsruher Institut für Technologie, DE
- Christian Schulz
KIT – Karlsruher Institut für Technologie, DE
- Anand Srivastav
Universität Kiel, DE
- Christian Staudt
KIT – Karlsruher Institut für Technologie, DE
- Veronika Strnadova
University of California – Santa Barbara, US
- Sivan Toledo
Tel Aviv University, IL
- Jesper Larsson Träff
TU Wien, AT
- Bora Ucar
ENS – Lyon, FR
- Panayot S. Vassilevski
LLNL – Livermore, US
- David Veith
Goethe-Universität Frankfurt am Main, DE
- Katharina A. Zweig
TU Kaiserslautern, DE