Research Statement

Darren Strash
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
strash@kit.edu

October 15, 2015

Viewpoint

Efficient algorithms play an integral role in every day computing. However, there is often a mismatch between theory and practice: efficiency in a theoretical model of computation does not necessarily imply efficiency in practice (and vice versa). One way to bridge the gap between theory and practice is to design algorithms that exploit the real-world properties of data sets—a strategy which has been described as viewing the world through "the algorithmic lens." In my own experience, this strategy has lead to strong results that benefit both theoreticians and practitioners.

Previous Work

It is through the algorithmic lens that I have viewed a number of problems in my research, with a focus on applications in large-scale network analysis and computational geometry. Most of my doctoral work, along with my most recent work, focuses on using real-world assumptions to develop a tighter run-time analysis than is possible with arbitrary inputs. These include 1.) linear-time algorithms for geometric graphs with few crossings (e.g., road networks), 2.) optimal encoding of coordinates for greedy routing in 3-connected planar graphs in the Euclidean plane, 3.) faster subgraph counting for sparse dynamic graphs, 4.) listing maximal cliques in near-optimal time for sparse graphs, 5.) updating dynamic planar point location data structures in sublogarithmic time for real-world geometric objects, and 6.) a linear-time algorithm for visualizing character interactions in storylines that have few characters.

In addition to tighter theoretical analyses, I have also investigated the impact of real-world assumptions on computational feasibility and tractability in practice. For instance: 1.) providing theoretical support for the "6 degrees of separation" of social networks under the assumption that people are members of few social groups, 2.) showing that we can quickly list all maximal cliques in large sparse networks using the memory available on a standard laptop, and, in my most recent work, 3.) showing that near-optimal independent sets can quickly be found in huge social networks, road networks, and Web graphs.

Methodology

My real-world viewpoint has lead me to focus on two key research methodologies: *interdisciplinary collaboration* and *algorithm engineering*, which often fit hand-in-hand.

During my doctoral work, I collaborated with sociologists to develop efficient algorithms for computing graph features in large social networks. Due to their size, it is often prohibitively expensive to compute interesting graph features, such as cliques, exactly in these these networks. However, during our collaboration, I developed and implemented the first algorithm to quickly list all maximal cliques in a large social networks when stored in a sparse representation. Previous efficient algorithms use the adjacency matrix representation for speed, and therefore are only suitable for small graphs. However, with the new algorithm, it is now possible to list all cliques in graphs with millions of vertices using a standard laptop.

After my doctoral work, I spent three years in research and development in Intel's Computational Lithography Group. There, my goal was to develop and maintain a massively-parallel geometric processing engine, which computes a manufacturable VLSI layout from a non-manufacturable one by compensating for physical limitations in optics. This required significant innovation and collaboration between programmers, algorithmicists, VLSI designers, and optics experts.

During the collaborative process, I pursue research using the *algorithm engineering* paradigm. Algorithm engineering gives equal treatment to both the theory and practice of algorithms, while using an application to drive algorithm development. Efficiency is developed in part by designing algorithms within realistic models of computation, and by analyzing input data to determine the structure of realistic inputs. However, algorithm engineering is also a feedback loop: instead of looking at theory or experiments in isolation, we experiment with algorithms to improve our understanding of the existing theory and vice-versa. Often, this strategy can help us explain the observed behavior of existing algorithms in practice, and develop new faster algorithms.

To further pursue algorithm engineering, and its application to massive data, I have been working at Karlsruhe Institute of Technology with Peter Sanders—one of the main proponents of the algorithm engineering paradigm. My most recent work applies the algorithm engineering paradigm to solving the maximum independent set problem on huge networks. In particular, we show that combining existing techniques from both the theoretical and experimental algorithms communities produces faster and more accurate algorithms than either technique alone.

Future Agenda

Moving forward, I will continue to pursue interdisciplinary collaboration as a means to develop efficient algorithms for solving interesting, well-motivated, real-world problems on massive data sets. It is my hope to continue treading the line between theory and practice by designing algorithms that are theoretically efficient, and also engineering algorithms that are fast in practice. Here is what I plan to tackle in the coming years:

Communication-Efficient Algorithms Though many efficient algorithms have been developed in the external memory and cache oblivious models of computation, the size of

real-world massive networks requires that real-world algorithms exploit parallelism. Therefore, I plan to focus a majority of my effort on developing techniques for designing efficient massively-parallel graph algorithms, using graph measures (e.g., diameter and betweenness centrality) and combinatorial optimization problems as primary problems. One of the major bottlenecks in parallel algorithmics is the *communication complexity*—how much communication is required between processing elements. My first goal is to investigate how we can use techniques from I/O-efficient algorithms to reduce the communication overhead for parallel algorithms. Graph measures and combinatorial optimization techniques have many applications in sociology and biology, and I believe research in this area would provide many opportunities for collaboration. Further, given the pervasive nature of similar problems in industry, I plan to actively pursue collaborations with companies such as Facebook and Google, who must process such large graphs on a continual basis.

Problem Subdivision for Massive Parallelism Related to big data is the concept of combinatorial scientific computing, where the goal is to develop efficient combinatorial algorithms, aiding high-performance computing by breaking down inputs into pieces that can be processed efficiently. Problems of interest in this domain include graph partitioning, graph coloring, computing graph separators, and computing connected components. To aid high-performance computing, I would like to develop efficient parameterized algorithms for solving graph partitioning and graph coloring on large real-world networks. Specific applications include identifying graph features to analyze networks for data mining, social interactions, protein-protein interactions, and reactions in physics and chemistry.

Open Science

Although my main goal is to produce high-quality research, it is also my intention to make my academic output freely available for anyone who wants it. For this reason, I post preprints of all of my articles on `www.arxiv.org`, an open repository for research articles. Further, I release any software that write for experiments under a free and open source (FOSS) license. I currently release all of my software under a GNU GPLv3.0 license, which requires any release of modified software to also release the modified code. All of my code is freely available on GitHub.

One of my long-term goals is to develop a general framework for experimental algorithmics on graphs. Several nice graph software packages exist; however, most of the experimental software I have seen from other researchers uses 100% self-developed code. It is my opinion that developing a common package is possible, and that having such a package would greatly reduce the time spent by researchers re-implementing and verifying other researchers' published results.