Helen Xu

1 Cyclotron Rd, B59-4024J
Berkeley, CA 94704
+1 (914) 462 7260
hjxu@lbl.gov
itshelenxu.github.io/
Google Scholar: `https://bit.ly/3vTLj7o`

Research Overview

I study theoretically and practically efficient parallel algorithms because now that Moore's law has ended, efficient algorithms are a principal way to gain performance for new applications. In particular, I have recently investigated sparse graph and matrix problems because of their ubiquity and susceptibility to mathematical analysis.

Current Position

Feb 2022 - Present **Grace Hopper Postdoctoral Scholar in Computing Sciences**, Lawrence Berkeley National Laboratory.
Host: Dr. Aydın Buluç.

Education

Feb 2022 **Ph.D. in Computer Science**, Massachusetts Institute of Technology.
Thesis: The Locality-First Strategy for Developing Efficient Multicore Algorithms.
Advisor: Prof. Charles E. Leiserson.

May 2016 **B.S. with Honors in Computer Science**, Stony Brook University.
Thesis: Write-optimized Skip Lists.
Advisor: Prof. Michael A. Bender.
Awards: Ranked first in the 2016 graduating class of over 100 students and received the Undergraduate Teaching Assistant Award (top 1% of TAs in the CS department).

Awards

2022 **Grace Hopper Fellowship in Computing Sciences** *for two years of postdoctoral study at Berkeley Lab.*

2022 **Argonne Training Program for Extreme-Scale Computing (ATPESC),** *a fully-funded two-week course on high-performance computing (awarded to about 80 researchers).*

2021 **Finalist for Best Student Presentation** *at ACDA 2021.*

2020 **Rising Stars Workshop** *held virtually at UC Berkeley.*

2020 **Chateubriand Fellowship** *for Spring 2020 at ENS Lyon.*

2016 **National Physical Science Consortium (NPSC) Graduate Fellowship,** *awarded for six years of graduate study.*

2016 **Undergraduate Teaching Assistant Award,** *awarded to around 1% of TAs in the department of Computer Science at Stony Brook University.*

2015 **NSF Research Experience for Undergraduates Supplement,** *awarded for one year of undergraduate research.*

2013-2016 **Christian Mata Scholarship for Excellence in Computer Science,** *awarded to the top ranked computer science student in every year at Stony Brook University.*

Publications

Published 14 peer-reviewed conference papers (10 full papers and 4 short papers) in top venues for algorithms and systems including SIGMOD, PODS, SPAA, and IPDPS.

Conference Publications

ALENEX '23 *Optimizing Search Layouts in Packed Memory Arrays*. Brian Wheatman, Randal Burns, Aydın Buluç, and **Helen Xu**. To appear at the SIAM Symposium on Algorithm Engineering and Experiments (ALENEX), 2023.

ESA '22 *When Are Cache-Oblivious Algorithms Cache Adaptive? A Case Study of Matrix Multiplication and Sorting*. Arghya Bhattacharya, **Helen Xu**, Abiyaz Chowdhury, Rathish Das, Rezaul A. Chowdhury, Rob Johnson, Rishab Nithyanand, Michael A. Bender. In European Symposium on Algorithms (ESA), 2022.

ACDA '21 *Multidimensional Included and Excluded Sums*. **Helen Xu**, Sean Fraser, and Charles E. Leiserson. In SIAM Conference on Applied and Computational Discrete Algorithms (ACDA), 2021.

SIGMOD '21 *Terrace: A Hierarchical Graph Container for Skewed Dynamic Graphs*. Prashant Pandey, Brian Wheatman, **Helen Xu**, and Aydın Buluç. In ACM SIGMOD International Conference on Management of Data (SIGMOD), 2021.

ALENEX '21 *A Parallel Packed Memory Array to Store Dynamic Graphs*. Brian Wheatman and **Helen Xu**. In SIAM Symposium on Algorithm Engineering and Experiments, 2021 (ALENEX).

APOCS '21 *Beyond Worst-case Analysis of Multicore Caching Strategies*. (in alphabetical order) Shahin Kamali and **Helen Xu**. In SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS), 2021.

SPAA '20 *Closing the Gap Between Cache-Oblivious and Cache-Adaptive Analysis*. (in alphabetical order) Michael A. Bender, Rezaul A. Chowdhury, Rathish Das, Rob Johnson, William Kuszmaul, Andrea Lincoln, Quanquan C. Liu, Jayson Lynch, and **Helen Xu**. In ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2020.

SPAA '18 *Cache-Adaptive Exploration: Experimental Results and Scan-Hiding for Adaptivity*. (in alphabetical order) Andrea Lincoln, Quanquan C. Liu, Jayson Lynch, and **Helen Xu**. In ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2018.

IPDPS '18 *A Fill Estimation Algorithm for Sparse Matrices and Tensors in Blocked Formats.* Peter Ahrens, **Helen Xu**, and Nicholas Schiefer. In IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2018.

PODS '17 *Write-Optimized Skip Lists.* (in alphabetical order) Michael A. Bender, Martin Farach-Colton, Rob Johnson, Simon Mauras, Tyler Mayer, Cynthia Phillips, and **Helen Xu**. In ACM Symposium on Principles of Database Systems (PODS), 2017.

Peer-Reviewed Short Publications

HPEC '20 *Work-Efficient Parallel Algorithms for Accurate Floating-Point Prefix Sums*. Sean Fraser, **Helen Xu**, and Charles E. Leiserson. In IEEE High Performance Extreme Computing Conference (HPEC), 2020.

SPAA '20 *Brief Announcement: Multicore Paging Algorithms Cannot Be Competitive*. (in alphabetical order) Shahin Kamali and **Helen Xu**. In ACM Symposium on Parallelism in Algorithms and Architectures (HPEC), 2020.

ICONS '19 *Dynamic Programming with Spiking Neural Computing*. (in alphabetical order) James Aimone, Ojas Parekh, Cynthia Phillips, Ali Pinar, William Severa, and **Helen Xu**. In International Conference on Neuromorphic Systems (ICONS), 2019.

HPEC '18 *Packed Compressed Sparse Row: A Dynamic Graph Representation*. Brian Wheatman and **Helen Xu**. In IEEE High Performance Extreme Computing Conference (HPEC), 2018.

Under Review

A Compressed Packed Memory Array with Fast Parallel Batch Updates. Brian Wheatman, Randal Burns, Aydın Buluç, and **Helen Xu**.

Optimizing Compression Schemes for Parallel Sparse Tensor Algebra. **Helen Xu**, Tao B. Schardl, Michael Pellauer, and Joel S. Emer.

BP-tree: Overcoming the Point-Range Tradeoff for In-Memory Key-Value Stores. **Helen Xu**, Amanda Li, Brian Wheatman, Manoj Marneni, and Prashant Pandey.

Research Experience

Feb 2022 - present **PASSION Laboratory** *Lawrence Berkeley National Laboratory*, Berkeley, CA.
Developing algorithms and data structures for sparse graph and tensor applications.
Mentor: Dr. Aydın Buluç.

Sep 2016 - Feb 2022 **Supertech Group** *Massachusetts Institute of Technology*, Cambridge, MA.
Designed, analyzed, and implemented parallel and cache-friendly algorithms for shared-memory multicores.
Mentor: Prof. Charles E. Leiserson.

Summer 2020 **Architecture Research Group** *NVIDIA Research*, Westford, MA.
Created and evaluated compressed data structures for sparse tensor computations.
Mentor: Dr. Michael Pellauer.

Spring 2020 **Resource Optimization: Models, Algorithms, and Scheduling Group** *ENS Lyon*, Lyon, France.
Studied algorithms for multi-resource job scheduling with theoretical guarantees.
Mentors: Professors Loris Marchal, Frederic Vivien, and Anne Benoit.

Summer 2019 **Systems and Networking Group** *Microsoft Research*, Cambridge, UK.
Developed and implemented algorithms for novel parallel machine learning accelerators.
Mentor: Dr. Nuno Lopes.

Summer 2016 **Discrete Mathematics and Optimization Group** *Sandia National Laboratories*, Albuquerque, NM.
Simplified and refined randomized I/O-efficient data structures.
Mentor: Dr. Cindy Phillips

Summer 2015 **Systems and Security Research Group** *Sandia National Laboratories*, Livermore, CA.
Designed and implemented a network event tracking database using an existing write-optimized fractal tree index.
Mentor: Dr. Tom Kroeger

Teaching

Qualifications

2021 **Kaufman Teaching Certificate Program (KTCP)** *Teaching and Learning Lab*, Massachusetts Institute of Technology, Cambridge, MA.
- Completed a semester-long course consisting of developing teaching skills via weekly workshops.
- Prepared and delivered a teaching demo on algorithms to a general audience and provided peer feedback.

Experience

2017 - 2018 **Teaching Assistant for Performance Engineering of Software Systems** *Department of Electrical Engineering and Computer Science*, Massachusetts Institute of Technology, Cambridge, MA.

- Received an average overall rating of 6.2 out of 7 points.
- Led weekly recitations (TA-led tutorial sessions) and office hours.
- Developed the course cloud infrastructure, including the course development environment and tools suite.
- Coordinated with instructors to write and grade homeworks and exams.
- Helped students with assignments on the course discussion forum.

2014 - 2015 **Teaching Assistant for Discrete Mathematics** *Department of Computer Science*, Stony Brook University, Stony Brook, NY.

Students Advised

Ph.D.

2018-present **Brian Wheatman**, Johns Hopkins University.
Hosted at Berkeley Lab in Summer 2022.

2020-present **Arghya Bhattacharya**, Stony Brook University.

Masters theses

Feb 2023 (expected) **Amanda Li**, MIT.
Overcoming the Point-Range Tradeoff in Key-Value Stores.

June 2020 **Sean Fraser**, MIT.
Optimizing Parallel Prefix Sums for Scientific Computing.

June 2019 **Brian Wheatman**, MIT (co-advised with Tim Kaler).
Image Alignment and Dynamic Graph Analytics : Two Case Studies of How Managing Data Movement Can Make (Parallel) Code Run Fast.

June 2019 **Stephanie Ren**, MIT (co-advised with Tao B. Schardl).
Vector-Aware Space Cuts in Stencil Computations.

Technical Talks and Presentations

Invited Seminars

Optimizing Dynamic Graph Processing on Multicores with the Locality-First Strategy

- Lawrence Berkeley National Laboratory, 2022.
- University of California, Berkeley, 2022.

Data Structure Design for Skewed Dynamic Graphs

- MIT Fast Code Seminar, 2021.
- Williams CS Colloqium, 2021.

Updatable Data Compression Formats for Hierarchical Fiber Abstraction

- NVIDIA Research, 2020.

A Fill-Estimation Algorithm for Sparse Matrices and Tensors in Blocked Formats

- Tel Aviv University, 2019.
- University of California, Berkeley, 2022.

Conference Talks

Multidimensional Included and Excluded Sums

- *Finalist for Best Student Presentation* at ACDA, 2021.

Beyond Worst-case Analysis of Multicore Caching Strategies
- APOCS, 2021.

Work-Efficient Parallel Algorithms for Accurate Floating-Point Prefix Sums
- HPEC, 2020.

Multicore Paging Algorithms Cannot Be Competitive
- SPAA, 2020.

Dynamic Time Warping in Strongly Subquadratic Time: Algorithms for the Low-Distance Regime and Approximate Evaluation
- ICALP, 2019 (filling in for Bill Kuszmaul).

Cache-Adaptive Exploration: Experimental Results and Scan-Hiding for Adaptivity
- SPAA, 2018.

Write-Optimized Skip Lists
- PODS, 2017 (Joint talk with Tyler Mayer).
- Guest lecture in MIT's advanced performance engineering course, 2017.

Posters

A Parallel Packed Memory Array for Dynamic Graphs
- Rising Stars Poster Session, 2020.

A Fill-Estimation Algorithm for Sparse Matrices and Tensors in Blocked Formats
- CSAIL Alliances Poster Session, 2018.

Performance Engineering Discussions

Software Performance Engineering at the End of Moore's Law
- Jane Street Symposium, 2020.

Leiserchess Codewalk
- Guest lecture in Performance Engineering of Software Systems (MIT 6.172), 2018.

Communication Workshops

NSF Graduate Research Fellowship Statements Workshop
- MIT EECS Communication Lab, 2020 and 2021.

Research Qualifying Examination Workshop
- MIT EECS Communication Lab, 2019 and 2021.

Professional Service

2022 **Artifact Evaluation Program Committee Member** *for PPoPP.*

2022 **Availability Reviewer** *for SIGMOD.*

2021-present **Program Committee Member** *for ALENEX (2022, 2023), SPAA (2022, 2023), SEA (2023), SC (Algorithms track, 2023).*

2018-present **Paper Reviewer** *for BigData (2023), JPDC (2022), ICPP (2021), ACDA (2021), WADS (2021), SPAA (2021, 2020), EuroPar (2020), SOSA (2020), SODA (2019, 2018), ESA (2019, 2022).*

2018 - 2022 **Communication Lab Advisor** *with the MIT Electrical Engineering and Computer Science Communications Lab.*

Communication Lab Page: `http://mitcommlab.mit.edu/eecs/`

- Reviewed over 50 papers, talks, statements, and CVs via peer coaching.
- Created and implemented a partnership with Professor Saman Amarasinghe's research group to jointly prepare conference presentations via weekly meetings.
- Prepared and led workshops for the NSF Graduate Research Fellowship application (2020, 2021) and the PhD research qualifying exam presentation (2019, 2021).
- Wrote articles about core technical communications topics, such as thesis proposals and paper abstracts.

2017 **Co-president** *of MIT Graduate Women in Course 6 (GW6).*

Webpage: `http://gw6.scripts.mit.edu/`

- Organized events several times per month for graduate women in the department, including professional development and social activities.

References

Prof. Charles E. Leiserson	Dr. Aydın Buluç
Department of Electrical Engineering and Computer Science	Computational Research Division
Massachusetts Institute of Technology	Lawrence Berkeley National Laboratory
cel@mit.edu	abuluc@lbl.gov
+1 (617) 642-6521	+1 (510) 325-9659
Prof. Michael A. Bender	Prof. Shahin Kamali
Department of Computer Science	Department of Computer Science
Stony Brook University	York University
bender@cs.stonybrook.edu	kamalis@yorku.ca
+1 (631) 987-8035	+1 (416) 727-2452

Research Statement - Helen Xu

My research centers around software performance engineering with a focus on locality and parallelism. I am the 2022 Grace Hopper Postdoctoral Fellowship in Computing Sciences at Lawrence Berkeley National Laboratory, which is an honor awarded to only one postdoc per year. I received my Ph.D. from MIT on "The Locality-First Strategy for Developing Efficient Multicore Algorithms" [1], which advocates for an algorithm development strategy that focuses first on the memory hierarchy before introducing parallelism. I have used this strategy to develop eight software systems that outperform their competitors. Notable among them are the parallel algorithms and data structures for sparse graph and matrix applications, including Terrace [2], a fast dynamic-graph-processing system that speeds up sparse graph algorithms by $2\times$ compared to the state of the art, and PHIL [3], an efficient fill-estimation algorithm for sparse matrices that achieves at least $2\times$ and up to $50\times$ speedup over the best existing algorithm for the problem.

I study theoretically and practically efficient parallel algorithms for multicores because now that Moore's law (i.e., the doubling of the number of transistors per computer chip every 2 years) has ended, efficient algorithms are a principal way to gain performance for new applications. Since Moore's law has ended and problem sizes continue to grow, researchers have shifted to software performance engineering, or optimizing software to make it run faster, to handle the ever-increasing volume of data (Leiserson *et al.* '20). Moore's law has driven improvements in computer performance and resulting economic progress. However, society can no longer rely on processor speed improvements from Moore's law for ubiquitous performance enhancements. Instead, I believe that future gains in computer performance on general-purpose multicores must come from tailoring specific applications to important features of the underlying hardware architecture.

Software performance engineering on multicores requires algorithm developers to design and optimize algorithms for two main salient hardware features: multiple cores and complex cache hierarchies. *Parallel* algorithms speed up execution by using the independent cores to perform multiple simultaneous operations. Furthermore, cache-friendly algorithms exploit *locality*, or the tendency of a program to access the same or similar data over time, to take advantage of the *cache hierarchy*, or multi-level memory stores with varying sizes and access speeds. On the theoretical side, developers use "work-span analysis" (Cormen *et al.* '09) and the "Disk-Access Machine model" (Aggarwal and Vitter '88) to analyze algorithms in the face of parallelism and caching, respectively. Additionally, achieving good empirical performance requires practical expertise about parallelism and locality. Parallelism and locality are notoriously difficult to optimize for independently and even more challenging to combine because they are often in tension [1].

My research vision is to make performance accessible to the average programmer without them having to worry about optimizing their code. To realize my vision, I will use software performance engineering to develop fast libraries for both general and application-specific use cases. Software libraries enable application developers to easily incorporate fast codes into their program with minimal effort from the user.

To create fast codes as concrete steps towards this vision, I contend that performance engineers should use a *locality-first strategy* [1–10] to create efficient parallel multicore algorithms. That is, they should first understand and exploit locality as much as possible before introducing parallelism. Optimizing first for locality results in the best serial algorithm, which enables the resulting code to reach peak efficiency after parallelization. Since optimizations for cache-friendliness may affect parallelism, introducing parallelism later minimizes the time a performance engineer spends rewriting their parallelization strategy.

I have collaborated with researchers across theory and systems to apply the locality-first strategy towards developing provably-good solutions that work well in practice. Performance engineers take advantage of two types of locality: *spatial locality*, or the tendency of programs to access contiguous memory locations, and *temporal locality*, or the tendency of programs to access the same memory location over time. A program may exhibit spatial and/or temporal locality depending on the problem. The main directions of my research to date study both of these types of locality:

- I optimized for spatial locality in data representations for sparse graph and matrix applications. These results have been published in four papers in top-tier conferences including SIGMOD and IPDPS [2–5].
- I analyzed the theoretical and practical interactions between temporal locality and parallelism. These results have been published in five papers in top conferences including SPAA and ESA [6–10].
- I have studied spatial/temporal locality and parallelism separately in an ad hoc fashion [11–14].

In the remainder of this statement, I will focus on the first two of these research directions and conclude with future work.

Optimizing for spatial locality in data representations

My studies [2–5] of sparse graph and matrix applications demonstrate that optimizing for spatial locality even at the cost of some parallelism can improve overall performance. Many sparse computations such as graph algorithms exhibit minimal temporal locality. Therefore, the focus of this line of study is on improving spatial locality by collocating data through cache-friendly data structure design. For example, my coauthors and I demonstrated [2, 5] that cache-friendly dynamic data structures can speed up sparse graph algorithms by 2× compared to the state of the art, even though the cache-friendly design poses challenges to parallelism.

Sparse graphs and sparse matrices appear in many fundamental applications that range from scientific computing to social networks. Sparse datasets have many more zeros than nonzeros, which allows performance engineers to write fast algorithms and efficient data structures with complexities that depend only on the number of nonzero entries. These data structures improve algorithm complexity by orders of magnitude because they allow direct computation over compressed data. Storing only the nonzeroes introduces additional complexity, however, because of the metadata required to store the locations of the nonzeroes. One of the main challenges for computing on sparse data is designing scalable and cache-friendly algorithms and data structures to store and process this metadata. These challenges present opportunities for designing algorithms and data structures that take advantage of modern hardware.

Collocating data for dynamic-graph processing

I have used the locality-first strategy to design and implement fast parallel dynamic-graph-processing systems [2, 5] that are optimized for spatial locality.

Many real-world graphs ranging from machine learning graphs (Piekniewski ‘09) to social networks (Faloutsos *et al.* ‘99) exhibit *graph irregularity*, or a sparse and skewed structure. These graphs have a skewed distribution of vertex degrees (Newman ‘05), where there are a few high-degree vertices and many low-degree vertices. Furthermore, these graphs are often *dynamic*: they change over time. For example, *dynamic-irregular* graphs arise naturally in social networks, computational biology, and the Internet. Much larger problems on the order of gigabytes and up to terabytes can be efficiently solved by exploiting irregularity, which is crucial for scaling applications to handle large graphs (Shun and Blelloch ‘13).

Existing static graph-processing systems optimized for graph irregularity (Chen *et al.* ‘19) achieve high performance and low space usage by preprocessing a cache-efficient graph partitioning based on vertex degree. In the dynamic setting, however, the whole graph changes over time. As a result, finding an optimal partitioning is not feasible in the presence of updates. Therefore, existing dynamic graph-processing systems use a “one-size-fits-all” representation, which pre-selects one type of data structure upfront for all vertices. The “one-size-fits-all” approach leaves performance on the table. To address this issue, my collaborators and I developed a framework based on hierarchical dynamic data structures that adapts to the underlying graph.

We introduced *Terrace* [2], a system for dynamic graphs that uses a cache-friendly design that adapts to naturally-occurring graph irregularity. Terrace uses a hierarchical data structure design to dynamically partition vertices based on their degrees and adapt to skewness in the underlying graph. Our evaluations show that Terrace supports faster batch insertions for batch sizes up to 1M when compared to Aspen (Dhulipala *et al.* ‘19), a state-of-the-art high-performance dynamic-graph-processing system. On graph query algorithms, Terrace is $1.7 - 2.6\times$ faster than Aspen. Surprisingly, in some cases, Terrace is even faster (up to 1.3× faster) than Ligra (Shun and Blelloch ‘13), a state-of-the-art static-graph-processing system, on graph queries. The reason for these performance gains is that Terrace experiences significantly fewer cache misses (in some cases, up to about 3× fewer than Ligra and about 6× fewer than Aspen) during graph queries because it exploits graph skewness for cache-friendliness.

One of the main components of Terrace is a *parallel Packed Memory Array* (PMA) data structure (Itai and Katriel ‘81), which enhances spatial locality by collocating all of the PMA’s data in memory at the cost of some parallelism [5]. Concurrently updating a PMA raises challenges because an update requires rewriting the entire structure in the worst case. However, my coauthor and I discovered [5] that 1) most of the updates only write to a small part of the structure and that 2) the worst case is highly parallel and cache-efficient. Therefore, the PMA is well-suited to concurrent updates on the whole. These results demonstrate the benefits of cache-optimized data structures in sparse graph computations.

Finding blocked structure in sparse matrices

I have applied the locality-first strategy in the study of blocked formats, which address memory bandwidth issues in sparse matrix workloads by improving spatial locality in the underlying data representation.

A common issue that arises in parallel implementations of sparse matrix algebra, such as sparse matrix-vector multiplication (SpMV), is limited memory bandwidth. This bandwidth issue is caused by irregular memory traffic due to the locations of the nonzeros. Researchers have shown that regularizing memory traffic with *blocked storage formats* (Vuduc *et al.* '02, Vuduc and Moon '05, Karakasis *et al.* '09) that take advantage of natural *blocked structure*, or clusters of nonzeroes, in sparse matrices can speed up SpMV by over $2\times$ on matrices with blocked structure. However, blocked storage formats must choose a block size that is carefully tuned to match the structure of a matrix to avoid unnecessary overhead.

To facilitate use of these blocked formats and alleviate the memory bandwidth issue, my coauthors and I introduced *PHIL* [3], a sampling algorithm with provable guarantees for estimating the *fill*, a metric for block-size quality. The fill has important applications in autotuning pipelines, but computing the fill exactly is too expensive. As a result, researchers developed OSKI (Vuduc *et al.* '05), an empirically fast-and-accurate heuristic for estimating the fill that samples matrix rows. In contrast to OSKI, the number of samples that PHIL requires is independent of the size of the input. This advantage of PHIL over OSKI is evident in our empirical evaluation: PHIL estimates the fill at least $2\times$ faster than OSKI on small matrices and $40-50\times$ faster on large matrices. Since PHIL has provable accuracy guarantees, it provides useful estimates of the fill even in pathological test cases where OSKI fails to estimate the fill within any useful error. PHIL's empirical success suggests broader potential for sampling techniques in autotuned numerical software.

Analyzing shared-memory algorithms with temporal locality

My research [6–10] in "beyond-worst-case" analysis of algorithms in shared memory, or memory that may be accessed simultaneously by multiple processes, demonstrates that cache-friendly algorithms are often mathematically and empirically good despite potential disruptions to their access patterns due to parallelism. Traditional worst-case analysis theoretically compares algorithms against an adversary on a single worst-case input, which may not reflect the common case in practice. In contrast, *beyond-worst-case analysis* (Roughgarden '20) articulates properties of real-world inputs and analyzes algorithms on inputs with those properties. I have used beyond-worst-case analysis to close the gap between theory and practice in cases where worst-case analysis fails to capture real-world properties of inputs such as temporal locality. For example, my coauthor and I showed [7] that the canonical Least-Recently-Used (LRU) algorithm theoretically matches or outperforms all other online algorithms for multicore cache replacement when accounting for locality.

Capturing locality in cache replacement for multicore processors

I used the locality-first strategy to separate multicore cache-replacement algorithms in the presence of naturally-occuring temporal locality via beyond-worst-case analysis. Every system with multiple cores sharing a cache needs to implement a cache-replacement algorithm, but there are few theoretical guarantees for the performance of these algorithms (López-Ortiz and Salinger '12).

As a first step, my collaborator and I showed [6] that all algorithms are arbitrarily far from optimal under worst-case competitive analysis. This theorem answers an open question from past work about the existence of a competitive algorithm in the negative. However, this first step left a gap between the theory and practice of shared caches. Specifically, LRU performs better than other online algorithms in practice because it exploits temporal locality (Albers *et al.* '02).

Our subsequent study [7] takes the first steps beyond worst-case analysis for multicore caching to resolve this gap between theory and practice. We introduced a new technique, called "cyclic analysis," that compares online algorithms on the entire space of inputs rather than a single worst-case input. Using cyclic analysis, we showed that LRU is the single best online algorithm on inputs with (temporal) locality in the multicore setting. These results validate the superiority of LRU in practice (Albers *et al.* '05) and demonstrate the potential of beyond-worst-case measures to mathematically capture practical differences between algorithms.

Smoothing analysis of cache-adaptive algorithms in shared memory

Additionally, I mathematically and empirically grounded the locality-first strategy by showing that "cache-oblivious algorithms" optimized for temporal locality perform well in shared memory beyond the worst case. Cache-oblivious algorithms achieve asymptotically optimal performance on all fixed cache sizes without knowledge of the memory size (Frigo *et al.* '99). However, most previous studies of cache-oblivious algorithms assumes that the amount of available cache is fixed during algorithm execution.

On multi-threaded and multicore systems, the amount of cache available to any single process constantly varies over time as other processes start, stop, and change their demands for cache. For years, experimentalists have developed heuristics and experimentally fast algorithms for major operations such as database sorts and joins (Pang *et al.* '93), but these algorithms are often vulnerable to worst-case inputs. To address this issue, researchers have studied *cache-adaptive* algorithms that gracefully handle changes in cache allocation (Bender *et al.* '14, '16). However, prior work [9] on cache adaptivity used worst-case analysis to separate algorithms, which may be overly pessimistic due to adversarial inputs tailored to algorithm structure.

To address this issue, my collaborators and I took the first steps beyond worst-case analysis of cache-adaptive algorithms both theoretically [8] via "smoothed analysis" (Spielman and Teng '01) and empirically [10] via beyond-worst-case workloads. Although cache-oblivious algorithms are optimal for fixed memory sizes, they may perform poorly on the worst-case memory profile (Bender *et al.* '14). On the theoretical side [8], we closed this gap between cache-obliviousness and cache-adaptivity mathematically with randomization of the worst-case profile. On the empirical side [10], we evaluated cache-oblivious algorithms under "oblivious memory profiles" that do not have knowledge of algorithm structure. We found that cache-oblivious algorithms capture almost all of the empirical benefits of cache adaptivity when the memory profile is not tailored to the algorithm structure. These results provide guidance for analyzing cache-adaptive algorithms beyond the worst case both in theory and in practice.

Future work

Going forward, I will use both the locality-first strategy and new methods to realize my research vision of making performance accessible. In the post-Moore era, computer scientists from all domains will need software performance engineering to solve any type of large-scale problem. I am excited to collaborate across the systems and theory communities to continue my research in the problem domains I have described as well as to find new applications to accelerate with algorithm design and engineering.

Compressed data representations. In the short term, I will further optimize the cache-friendly data structures I have built by adding compression to the Packed Memory Array (PMA) in Terrace [2] and PPCSR [5]. Compression improves spatial locality in data representations (Smith '97) but unfortunately introduces challenges to parallelization by serializing the computation. The PMA uses a constant factor of extra spaces by design, so adding compression resolves one of the major drawbacks to using a PMA for dynamic-graph processing as well as for general applications. In fact, compression is necessary to fit today's largest graphs in memory on a single multicore (Dhulipala *et al.* '18).

Applicability of dynamic-graph data structures. I plan to expand the current suite of algorithms on top of the dynamic graph data structures I have developed [2, 5]. from the Ligra interface to the Graph Based Benchmark Suite (GBBS) (Dhulipala *et al.* '19), which encompasses a much larger suite of algorithms. The dynamic graph data structures that I have developed support a simple set of primitives such as map (parallel iteration) in order to implement the Ligra interface (Shun and Blelloch '13), which enables algorithm engineers to express many classical graph algorithms such as breadth-first search and PageRank but stops short of more complex algorithms such as bucketing and clustering (Dhulipala *et al.* '18). This broader suite of algorithms will not only expand the applicability of the underlying data structures but also provide a more comprehensive view of the performance differences between different types of data structures.

Beyond-worst-case analysis with prediction. I aim to further theoretically characterize the benefits of temporal locality by studying *prediction* (Lykouris and Vassilvitskii '18, Rohatgi '20) for multicore cache replacement. During my Ph.D., I took the first step in beyond-worst-case analysis of multicore cache replacement via "cyclic analysis" to account for locality in real-world inputs [7]. Prediction is another technique for beyond-worst-case analysis. Under the framework of algorithm design with predictions, algorithms use some (potentially erroneous) predictions to improve their solution quality (for online algorithms) or their running time (for offline algorithms). Augmenting algorithms for multicore cache replacement with predictions would enable them to take advantage of predictable patterns in real-world inputs (Denning '80).

Locality-first strategy on other computational platforms. In the longer term, I plan to investigate the locality-first strategy beyond shared-memory multicores for other parallel architectures such as GPUs and distributed systems. GPUs contain multiple levels of hierarchical memory with increasing access cost in lower levels. Similarly, communication due to data movement is a significant bottleneck in distributed systems. I believe that the locality-first strategy will result in efficient codes beyond multicores because data movement is also critical to performance in other computational models.

References

[1] H. Xu, “The locality-first strategy for developing efficient multicore algorithms,” Ph.D. dissertation, Massachusetts Institute of Technology, 2022.

[2] Prashant Pandey, Brian Wheatman, **Helen Xu**, and Aydın Buluç, “Terrace: A Hierarchical Graph Container for Skewed Dynamic Graphs,” in *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2021.

[3] Peter Ahrens, **Helen Xu**, and Nicholas Schiefer, “A Fill Estimation Algorithm for Sparse Matrices and Tensors in Blocked Formats,” in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018.

[4] Brian Wheatman and **Helen Xu**, “Packed Compressed Sparse Row: A Dynamic Graph Representation,” in *IEEE High Performance Extreme Computing (HPEC) Conference*, 2018.

[5] ——, “A Parallel Packed Memory Array for Storing Dynamic Graphs,” in *SIAM Symposium on Algorithm Engineering and Experiments (ALENEX)*, 2021.

[6] (in alphabetical order) Shahin Kamali and **Helen Xu**, “Brief Announcement: Multicore Paging Algorithms Cannot Be Competitive,” in *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2020.

[7] ——, “Beyond worst-case analysis of multicore caching strategies,” in *SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.

[8] (in alphabetical order) Michael Bender, Rezaul Chowdhury, Rathish Das, Rob Johnson, William Kuszmaul, Andrea Lincoln, Quanquan C. Liu, Jayson Lynch, and **Helen Xu**, “Closing the Gap Between Cache-Oblivious and Cache-Adaptive Analysis,” in *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2020.

[9] (in alphabetical order) Andrea Lincoln, Quanquan C. Liu, Jayson Lynch, and **Helen Xu**, “Cache-Adaptive Exploration: Experimental Results and Scan-Hiding for Adaptivity,” in *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

[10] Arghya Bhattacharya, **Helen Xu**, Abiyaz Chowdhury, Rathish Das, Rezaul A. Chowdhury, Rob Johnson, Rishab Nithyanand, and Michael A. Bender, “When Are Cache-Oblivious Algorithms Cache Adaptive? A Case Study of Matrix Multiplication and Sorting,” in *European Symposium on Algorithms (ESA)*, 2022.

[11] (in alphabetical order) Michael Bender, Martin Farach-Colton, Rob Johnson, Simon Mauras, Tyler Mayer, Cynthia Phillips, and **Helen Xu**, “Write-Optimized Skip Lists,” in *ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, 2017.

[12] **Helen Xu**, Sean Fraser, and Charles E. Leiserson, “Multidimensional Included and Excluded Sums,” in *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA)*, 2021.

[13] Sean Fraser, **Helen Xu**, and Charles E. Leiserson, “Work-Efficient Parallel Algorithms for Accurate Floating-Point Prefix Sums,” in *IEEE High Performance Extreme Computing (HPEC) Conference*, 2020.

[14] Brian Wheatman, Randal Burns, Aydın Buluç, and **Helen Xu**, “Optimizing Search Layouts in Packed Memory Arrays,” in *SIAM Symposium on Algorithm Engineering and Experiments (ALENEX)*, 2023.

Teaching Statement - Helen Xu

My teaching philosophy is to provide students with widely-applicable skills and hands-on experience so that they can independently solve real-world problems. Specifically, I am excited to train well-rounded future computer scientists with not only strong technical skills but also strong communication skills. My teaching approach combines lectures with hands-on projects to give students opportunities to apply the material.

Summary of experience. My experience in classroom teaching spans theory and systems. At MIT, I served twice as a TA for *Performance Engineering of Software Systems*[1] (about 120 students) during my Ph.D. and received an overall rating from student evaluations of 6.2 out of 7 points. As the head TA, I learned how to manage the student teaching staff in large classes. Additionally, I have served as a teaching assistant in both small (about 20 students) and large discrete mathematics courses (about 200 students) in my prior undergraduate teaching experience at Stony Brook University. In 2016, I was awarded the departmental Undergraduate Teaching Assistant award, which recognizes about 1% of TAs each year.

To further improve my teaching skills, I completed the Kaufman Teaching Certificate Program[2] (KTCP) at MIT. The KTCP is a semester-long workshop series for academics interested in teaching that covers topics including course design, active learning techniques, and providing effective feedback.

Outside of the classroom, I have advised three completed Masters theses at MIT and am currently currently advising one Masters student at MIT and two Ph.D. students: one at Johns Hopkins University and another at Stony Brook University. I have also peer coached over 50 students in technical communication.

Classroom teaching

My teaching approach centers around encouraging students to take as much of a role in their learning as possible so that they become independent problem solvers. Examples include:

Active learning. When preparing a course, I will make sure to include as many opportunities to interact with students as possible to engage them in the material. Studies have shown that active learning[3], or including students in the construction of their own knowledge, leads to more effective learning and better outcomes. For example, in a performance engineering course, I will incorporate questions for the students such as asking them to predict how an optimization will affect a program. Outside of lecture, as a TA for the performance engineering course at MIT, I developed weekly hands-on tutorials where the students gather in small groups with a TA to work through exercises related to that week's topics. This method also enabled the students to ask questions in real time as they apply the material from the lecture to actual code. The students found the exercises useful and often spent time finishing them even outside of the tutorial hours.

Instructional scaffolding. When teaching any new topic, I will break up the topic into chunks that start with previously known material and build up to newer ideas. This method is called *instructional scaffolding*[4] and has been shown to facilitate efficient learning by providing support when concepts are first being introduced. I was first introduced to scaffolding when preparing problem sets and exams for the performance engineering course at MIT. For example, I broke up the analysis of recursive cache-friendly algorithms into stages that would later be combined to reach the final asymptotic bound. Dividing the problem into multiple steps enabled the students to build on previous knowledge and identify where they were applying new material.

Project-based coursework. I will use projects when applicable to connect theory and practice. For example, in software performance optimization, I believe that hands-on implementation is the best way to learn how to apply different optimizations and see their effect on the results. I applied this principle in the performance engineering course at MIT, where I helped to develop the final project of optimizing and then parallelizing a chess engine. The project helped students to learn to first perform as many serial optimizations as possible before introducing parallelism. Furthermore, the month-long project encouraged students to revise their code and make additional optimizations after the early stages.

[1] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-172-performance-engineering-of-software-systems-fall-2018/

[2] https://tll.mit.edu/programming/grad-student-programming/kaufman-teaching-certificate-program/

[3] https://360learning.com/blog/active-learning/

[4] https://www.educationcorner.com/scaffolding-education-guide.html

Research mentorship

I look forward to starting my own research group and helping both graduate and undergraduate students cultivate an interest and sense for research. My experience in mentoring masters and Ph.D. students has helped me to develop strategies to help students become confident in their research.

Developing communication skills via active collaboration. As a mentor, I support students not only in making technical contributions but also in improving their communication skills. In the theses I have supervised, I have noticed that students are often less familiar with technical communication because it is not the focus of their coursework. For example, my advisee Sean Fraser was one of the top students in the performance engineering course at MIT. During our collaboration, I observed that Sean had less experience with technical writing than implementation, so I stepped into a hands-on collaborator role during the writing phase of the project to provide more guidance in paper structure and execution. I used active discussion and pair writing to help him build confidence in expressing his ideas. After working closely together on writing, I noticed that Sean was able to organize and present his results more clearly. As a result of my mentoring experience, I learned valuable lessons about how to interact with students to help them develop their skills. As a result, Sean and I published two conference papers from his master's thesis.

Encouraging questions. I actively use myself as an example by asking questions and demonstrating that research is not about having all the answers upfront. I remind my mentees that I am available to answer any questions they have between our meetings and that I am here to try to unblock them. This approach has worked especially well with my current masters student, Amanda Li. By asking questions between our one-on-one meetings, she has been able to participate more actively in group project meetings.

Peer coaching for technical communication

I am also committed to helping students develop strong communication skills in addition to their technical skills. Towards this goal, I served as a Communication Lab[5] (Comm Lab) advisor at MIT from 2018-2022. As a Comm Lab advisor, I provided peer coaching services to over 50 students on scientific communication tasks including statements, papers, and presentations. When anonymous feedback was solicited from students that I worked with, they have responded with comments like "[she gave] really good feedback about what was missing from my writing!" and "She advised me how I should guide a reader top-down toward my research. It was very helpful."

Asking guided questions. Instead of reviewing a document and giving directives for changes, I ask questions to unearth the story behind the student's work. For example, most recently, Michael Wang, an early-stage Ph.D. student at MIT, met with me via the Comm Lab to work on his NSF Graduate Research Fellowships Program (GRFP) statements. During our sessions, I asked high-level questions to prompt him to think about and speak to his audience who, like me, may not be experts in his area. Additionally, I asked questions to connect the points in his personal statement into a compelling narrative. I did not give him answers to those questions, but I encouraged him to think about them and incorporate his answers into his writing. He set up several coaching sessions with me to iterate on his statements and expressed that he found the feedback helpful. After our individual coaching sessions, Michael was selected for the NSF GRFP in 2022.

Teaching interests

I believe that both strong technical and communication skills are necessary for effective team problem-solving and am excited to teach both.

I enjoy teaching and am excited to teach undergraduate-level courses in algorithms, data structures, and performance engineering. I am qualified to teach other undergraduate courses such as programming, discrete mathematics, and theory of computation. Additionally, I look forward to developing and teaching graduate-level courses in parallel algorithms and data structures as well as seminars related to my research interests spanning parallel algorithm design, analysis, and engineering.

I also hope to share my experience in scientific communication through workshops, seminars, or even a potential course for graduate students. For example, I have held workshops on scientific presentations and NSF fellowship proposals through the MIT Communication Lab.

[5] `https://mitcommlab.mit.edu/eecs/`

Diversity, Equity, and Inclusion Statement - Helen Xu

Computer science (CS) in post-secondary education suffers from a representational gap: in 2020, 21% of Bachelor's degrees in CS were awarded to women and 15% to members of underrepresented ethnic groups[1]. For underrepresented groups who stay in computer science at the undergraduate level and beyond, being dramatically outnumbered makes it difficult to find a support network with the same background and societal expectations and fuels feelings of impostor syndrome[2].

My goal as a professor is to address these challenges to diversity, equity, and inclusion in two primary directions: (1) building support networks for underrepresented groups, and (2) facilitating collaboration in teaching and research. During my Ph.D., I gained experience with building support networks as co-president of Graduate Women in Course 6 (GW6) at MIT in 2017. Furthermore, I worked towards facilitating collaboration as a teacher and researcher. I plan to continue actively building connections and collaborations as a faculty member.

Building support networks. I believe that one of the key challenges that underrepresented students in CS face is the difficulty in making connections with other students with similar backgrounds because of their sparsity. For example, a woman graduate student might not have other women in her immediate research group or day-to-day interactions, which can compound feelings of isolation. I believe that social events for underrepresented groups in the department can address the issue by providing an informal space to make connections with other similar students. To help organize events for social and professional development, I served as co-president of Graduate Women in Course 6 (GW6) at MIT in 2017. GW6 is a student group of around 100 women PhD students in Electrical Engineering and Computer Science. One example GW6 event that I ran was a workshop on elevator pitches. The workshop helped graduate women explain their research to a broader audience and connect with other women in the department.

As a faculty member, I will continue to build support networks by working with groups for underrepresented students in the department to provide mentorship. For example, as co-president of GW6, I organized informal networking meetings (e.g., breakfasts, tea times) to connect women professors and GW6 members. These meetings led to mentorship connections between students and faculty who might not have met otherwise.

Facilitating collaboration. Another main challenge that underrepresented groups in computer science face at the undergraduate and graduate level is the "collaboration problem[3]." Fruitful collaborations are integral to success in coursework as well as research, but underrepresented groups encounter challenges because of the way they are perceived in a technical context. For example, women often have a more limited pool of project partners in team-based coursework because of implicit bias and weaker social ties within CS.

During my time as a TA for *Performance Engineering of Software Systems* at MIT, I facilitated smooth collaborations by reaching out to all of the students in my recitation to discuss how they felt about their project groups. Although the groups were mostly self-selected, students that did not have pre-selected partners coming into the class were assigned randomly. If a student reported problems in their group, I would discuss further with them and potentially mediate a meeting with the entire group.

Furthermore, I take the role of an active research mentor by making myself available for questions outside of scheduled meeting times. I encourage my mentees to ask for help as soon as possible rather than waiting for our next meeting. If at an appropriate stage in the project, I also prefer to take an active collaborator role, where I will set aside more time to pair-write with my research mentees, especially if they do not have much experience with technical writing.

In order to address the collaboration problem[4] in any class I teach, I plan to facilitate discussion about collaboration with the entire class since collaboration affects everyone, not just underrepresented groups. I will also check in with students throughout the course to make sure their collaborations are going smoothly. In my own research group, I will actively reach out to the students I work with to make sure they have enough collaborators in different roles.

Conclusion. My goal going forward is to continue my efforts to (1) organize and be part of support networks for underrepresented groups in CS and (2) promote collaboration in both my teaching and research to create a friendly and welcoming environment. This effort will enable us to develop a diverse set of talent and to make computing accessible for people with all kinds of backgrounds and needs.

[1] https://cra.org/wp-content/uploads/2021/05/2020-CRA-Taulbee-Survey.pdf

[2] https://ucsdguardian.org/2019/05/12/impostor-syndrome-computer-science/

[3] http://jxyzabc.blogspot.com/2016/11/why-we-need-to-talk-about-collaboration.html

[4] http://jxyzabc.blogspot.com/2016/12/follow-on-discussion-about.html