Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms   Monday, April 4, 2016
Ankur Moitra

Final Project

Proposal Due: Monday, April 18, 2016

Report Due: Wednesday, May 11, 2016

Key Points

Collaboration: You can work either individually or in groups of up to 3 students.

Proposal: 1-2 paragraph email to Professor Moitra and the TAs describing your project and listing references to relevant papers. One email per group.

Final Report: $\leq$ 10 page project report, with optional appendix of unlimited length including extra code, figures, proofs, exposition, etc. One per group.

Resources: Although not required, we encourage you to discuss project ideas before or after the proposal deadline with Professor Moitra or one of the TAs. Please email us to set up a time to chat.

Types of Projects

The goal of this project is to independently explore an advanced theoretical topic in algorithm design. This can be done in a number of ways, but we expect projects to fall into roughly 3 categories.

Reading Project: Find one or several research papers on an algorithm, class of algorithms, or algorithmic topic that you find interesting. The papers should be new enough or challenging enough so that their main results have not already been synthesized for books or lecture notes[1]. Write an exposition of the theoretical results.

This exposition should not simply compile theorems nor should it reprove every step of an analysis in detail. Instead, try to imagine you're preparing a lecture or series of lectures for our class. You want to explain and motivate the key ideas and techniques in the papers you study. You might include expositions of required background material, simplified proofs, motivating examples, open questions, etc. Your target audience is a student of this class and you'll be graded on the accuracy and clarity of your exposition.

[1] With a few exceptions: some of the suggested topics have been simplified in notes, books, and lectures before. However, they are sufficiently important that a fresh exposition would be interesting and valuable.

Theoretical Research Project: Find an open research question in algorithms and attack it. You might present a new algorithm for a studied problem or adapt a known algorithm or technique to an interesting new problem. You might improve the analysis of an existing algorithm or show that it can be improved for a certain special (and interesting) class of inputs. You might prove additional properties of an algorithm – e.g. that it can be implemented in small space or works under relaxed problem assumptions.

As with any research project, you'll need to prepare for the possibility that you don't solve the problem you set out to solve! This is perfectly fine – in that case you can present partial results, a clear exposition of the papers you've been studying, a discussion of promising avenues for future work, examples of hard input cases for a new analysis, etc. Imagine that you're writing a primer for anyone getting started on the problem you set out to solve.

Course projects are a great way to get started on longer term research, so we definitely encourage these sorts of projects. If you have a topic you like but are having trouble finding a concrete question or direction in the area, please see us. If we can't help, it is likely that we can point you to someone else in Cambridge who can.

Implementation Project: Yes, even though this is a theory course, your final project can involve coding! In fact, implementation is often one of the best ways to truly "learn" an algorithm and can provide invaluable insight into problem assumptions, potential optimizations, and new directions.

However, you should not simply implement one of the algorithms we discussed in class and show that it works. Instead, try to guide your project with an underlying question. Do two alternative algorithms for the same problem perform differently on different types of data? Do experiments suggest that the theoretical analysis of an algorithm could be tightened? Or do they rule out a tighter analysis or provable guarantees for a known heuristic? You can also propose and test optimizations that improve performance for some inputs, even if they do not provide any theoretical runtime improvements.

Implementation projects can be very time consuming, so make sure you choose a manageable algorithm or piece of an algorithm to work with. Your report should include a discussion of your goals, background material on the algorithm(s) implemented, experimental data, and a discussion of conclusions and future work. If the algorithm is not especially well known or studied, it might be a good idea to give some exposition on the main techniques involved.

FAQs

Can my project be related to my current research?

Yes, but we do not want you to present something you have already been working on. If you work in a field other than algorithms, you could apply ideas or techniques from this class (or other algorithmic papers) to a specific problem in your field. If your research involves algorithms already, you could explore an entirely new approach or use the project as motivation to learn a different sub-area. If you have any doubt about whether or not a project overlaps too much with your current work, please talk to Professor Moitra or a TA.

None of the suggested project topics are grabbing my interest. Where can I look for alternative topics and papers?

One good option is to look through the proceedings of recent algorithms conferences. If you want to refine your search even further, look for award winning papers[2] at these conferences or papers that seem well cited on Google Scholar. Do be aware of paper difficulty. Some papers will be written for a general theoretical computer science audience while some will cater to researchers in a very specific sub-area. These papers might be tough to read without a lot of background and prior knowledge. Again, check with us if you have any doubts. Some good starting places are:

- ACM-SIAM Symposium on Discrete Algorithms (SODA) – theoretical algorithms conference, with many relevant papers.
- ACM Symposium on Theory of Computing (STOC) – theoretical computer science conference, with many algorithms papers.
- IEEE Symposium on Foundations of Computer Science (FOCS) – theoretical computer science conference, with many algorithms papers.
- International Colloquium on Automata, Languages and Programming (ICALP) – theoretical computer science conference, with many algorithms papers.
- International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and International Workshop on Randomization and Computation (APPROX/RANDOM) – algorithms conference with a number of theoretical papers.
- European Symposia on Algorithms (ESA) – algorithms conference with a number of theoretical papers.
- Advances in Neural Information Processing Systems (NIPS) – machine learning conference with an increasing number of papers on algorithms for data problems. Do be aware that we want you to study an algorithm with some sort of theoretical guarantee.
- International Conference on Machine Learning (ICML) – machine learning conference with a number of papers on algorithms for data problems.
- Conference on Computational Learning Theory (COLT) – machine learning conference with a number of papers on algorithms for data problems.

These are only the broadest relevant conferences. If you're interested in a more specific area, we can point you to other venues.

I am having trouble finding/accessing papers online.

Many newer computer science papers are available for free online through `arxiv.org` or author websites. Google Scholar is a good place to check for links to freely available PDFs.

[2] See `http://jeffhuang.com/best_paper_awards.html` for a list of conference best paper awards.

If you're on an MIT network, you can also access papers through publisher websites like dl.acm.org, ieeexplore.ieee.org, epubs.siam.org, link.springer.com, etc. However, if you live/work off-campus, you'll find these sites unavailable. You'll need to access them through the MIT Library Proxy server. See for instructions.

Specific Project Ideas

Below we've included a list of potential project topics. We've also included citations to seminal papers or standard references on each topic. These papers are probably good starting points for reading projections. However, if you're doing a research project, you might have search a bit wider to find tackle-able open problems. One good approach is to look up the paper below on Google Scholar and then look at more recent papers that cite them.

Streaming Algorithms, Sketching, Dimensionality Reduction, etc.

- This is a very active area of research. There's a cool list of open research problems at: http://sublinear.info/

Linear Sketches for Graph Problems

- Ahn, Guha, McGregor, 2012, "Analyzing Graph Structure via Linear Measurements".
- Ahn, Guha, McGregor, 2012, "Graph Sketches: Sparsification, Spanners, and Subgraphs".

Frequent Directions Sketches: Misra-Gries for Matrices

- Ghashami, Liberty, Phillips, Woodruff, 2015, "Frequent Directions : Simple and Deterministic Matrix Sketching"

Subspace Embeddings / Johnson-Lindenstrauss for Fast Linear Algebra

- Sarlós, 2006, "Improved approximation algorithms for large matrices via random projections"
- Clarkson, Woodruff, 2013, "Low Rank Approximation and Regression in Input Sparsity Time"
- Nelson, Nguyen, 2013, "OSNAP: Faster Numerical Linear Algebra Algorithms via Sparser Subspace Embeddings"

Even Faster JL / Restricted Isometry Property

- Khramer, Ward, 2010, "New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property"

- Haviv, Regev, 2016, "The Restricted Isometry Property of Subsampled Fourier Matrices"

Metric Embeddings

- Bourgain, 1985, "On Lipschitz embeddings of finite metric spaces in Hilbert space"
- Linial, London, Rabinovich, 1995, "The geometry of graphs and some of its algorithmic applications"

Metric Embeddings II – Tree Embeddings

- Fakcharoenphol, Rao, Talwar, 2004, 'A tight bound on approximating arbitrary metrics by tree metrics"
- Mendel, Schob, 2009, "Fast C-K-R Partitions of Sparse Graphs"
- Abraham, Neiman, 2009, "Using Petal-Decompositions to Build a Low Stretch Spanning Tree"

Locality Sensitive Hashing / Nearest Neighbors

- Har-Peled, Indyk, Motwani, 2012, "Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality"
- Charikar, 2002, "Similarity Estimation Techniques from Rounding Algorithms"

Extended Formulations for Combinatorial Optimization

- Conforti, Cornuéjols, Zambelli, 2009, "Extended formulations in Combinatorial Optimization"
- Goemans, 2015, "Smallest Compact Formulation for the Permutahedron"

Matroids

Primal-Dual Approximation Algorithms

- Goemans, Williamson, 1997, "The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems"
- Devanur, Jain, Kleinberg, "Randomized Primal-Dual Analysis of RANKING for Online Bipartite Matching"

Iterative Rounding for Approximation Algorithms

- Jain, 1998, "A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem"

- Singh, Lau, 2007, “Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal”

Online Convex Optimization / Follow-the-leader Methods

- Kalai, Vempala, 2005, “Efficient algorithms for online decision problems”
- Hazan, Kalai, Kale, Agarwal, 2006, “Logarithmic Regret Algorithms for Online Convex Optimization”

Accelerated Gradient Descent

- Nesterov, 2004, “Introductory Lectures on Convex Optimization”
- Bubeck, 2014, “Convex Optimization: Algorithms and Complexity”
- Bubeck, Lee, Singh, 2015, “A geometric alternative to Nesterov’s accelerated gradient descent”
- Allen-Zhu, Orecchia, 2014, Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent

Theory of Stochastic Gradient Descent

- Hazan, Kale, 2011, “ Beyond the regret minimization barrier: An optimal algorithm for stochastic strongly-convex optimization”
- Rakhlin, Shamir, Sridharan, 2011, “Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization”
- Johnson, Zhang, 2013, “Accelerating stochastic gradient descent using predictive variance reduction”

Approximate Graph Coloring

- Karger, Motwani, Sudan, 1998, “Approximate Graph Coloring by Semidefinite Programming”

Cheeger’s Inequality / Spectral Graph Partitioning

- Good starting place for references: `http://cstheory.stackexchange.com/questions/3245/papers-to-credit-for-spectral-partitioning-of-graphs`
- Spielman, Teng, 2007, “Spectral Partitioning Works: Planar graphs and finite element meshes”

Approximate Counting

- Dyer, 2003, “Approximate counting by dynamic programming”

Community Detection

- Feige, Kilian, 1998, "Heuristics for finding large independent sets, with applications to coloring semi-random graphs"
- Abbe, Bandeira, Hall, 2016, "Exact Recovery in the Stochastic Block Model"