CS 583 Project Guidelines*

Instructor: Chandra Chekuri

You will be doing a final project in CS 583; 20% of your grade for this course will derive from this project. Ideally, this will be a project on a topic that is of great interest to you, and better yet, if it's relevant to your work/research in some way.

The plan is for you to pick a paper on approximation algorithms from the theoretical computer science literature, something that appeared in a conference (such as FOCS, STOC, SODA, ICALP, ITCS, APPROX, etc), or something from one of the main journals (e.g., SICOMP, JACM, Mathematics of Operations Research). Ideally, the paper should be relatively recent, say published in the last 20 or so years. And of course, the paper should not be something I will be covering in class.

You are encouraged to do the project in groups of 2 students (but no more).

Project Options

You can choose from one of the following 4 types of project.

- **Option 1:** Write a summary of the paper, containing the following items:
  - **Introduction:** problem definition, motivation, applications, and a brief summary of prior work on the problem.
  - **Main contributions:** the main results of the paper, the algorithm or technique and the implications of the results (e.g., improvement in approximation ratio, new problem class addressed, etc).
  - **Overview of the proof:** Give a high-level overview of the proof and the main ideas. What is the novelty of the paper which makes it different from the previous work?
  - **Details of the proof:** Go over the main steps and the main lemmas of the proof, explaining it in your own way. Make an effort to add value to the published version: more intuition, extra examples, details filled in, clarifying figures, clearer exposition, etc.
  - You may **not** copy/paste lemmas/proofs from the paper.
  - If the paper is long and has multiple non-trivial results it is reasonable to focus on the main one or two. Please feel free to consult with the course staff.
- **Option 2:** Empirical study. In most theory papers, there is no empirical evaluation. If you choose this option, you should plan on implementing the algorithm proposed in the paper and running it against some test cases and possibly alternative algorithms. You may generate your own test cases or you may use actual data from real world applications. It would be ideal if you compare the performance of the algorithm with a "greedy" algorithm or any reasonable baseline algorithm. Please write a report describing the algorithm proposed in the

*Adapted from Anna Karlin's course.

paper, a description of your experimental setup and the details of your findings. How does the empirical approximation ratio compare with the theoretical guarantee?

In both cases, you will be submitting your code and your test data.

- **Option 3:** Application to a different research area. In this case, the goal is for you to relate the paper to your own field of study. For example, you may modify the algorithm suggested in the paper and add a heuristic on top of it and run it against a real world problem that interests you. In your report, please describe the algorithm in the paper and your modifications. Then, explain how this can be used to solve a problem in your research area or to model a phenomenon that you are interested in. Then, describe the test data and the performance of your proposed solution for those test cases. Theoretical justification for your proposed solution would make this a great project. Your project could also be a purely theoretical one related to the application area.

- **Option 4:** Open ended research on a particular topic/problem. You may want to solve something concrete and have a problem or ideas for a problem. You should articulate the problem clearly and what you plan to work on. Feel free to ask the course staff for problems and advice.

In all four cases, you should make an effort to include a discussion of interesting open problems related to your topic (whether theoretical or applied).

Deadlines and evaluation

- Each group should turn in a 2-paragraph preproposal by **Thursday, April 2nd** on Gradescope. This pre-proposal must include the citation for the paper you will be working from. Your pre-proposal should also include a description of the type of project you will be doing from among the options listed above. If you are planning to do options 2 or 3, please provide some details on your plan.

- The final 7–10 page paper on your project will be due **Saturday, May 9th**. You will submit it on Gradescope but also post it for other students to see on Ed. If you do not wish to make the paper available for others to see, you can sign up for a half hour presentation that is open to students from the course.

- The paper will be evaluated on clarity, correctness, completeness, and depth. Your paper should be easy to read and not assume any background beyond what we've done in class (or what was assumed for class). Assume that the reader will have no knowledge about the topic and you are trying to give them intuition and a great entry point to it.

Possible topics

There are so many exciting topics that we do not have time to cover during the semester and every topic that we are covering is being done only at a superficial level. So on pretty much any topic we are touching upon, there are many interesting papers we are not covering that you could look into. A list of recent papers on approximation algorithms from major venues is available on the course webpage.

Resources - courses, etc

Here are some courses/lecture notes/surveys etc that might help you get started on a topic.

- Books on the course website.
- TCS+ talks: This is a collection of online seminars in theoretical computer science. A great entrée into reading a paper would be watching one of these talks and then diving into the associated paper.
- Previous editions of this course and related courses at other universities (see course webpage for links).

Resources - possible broad topics

Here are some topics you could explore:

- Covering and packing problems: (knapsack cover inequalities in approximation, maximum independent set in special classes of graphs, geometric instances, contention resolution schemes for packing)
- Scheduling (many problems)
- Clustering (k-median, k-means, k-center, facility location), several recent papers on k-means
- Network design (improved approximations for Steiner tree, Steiner forest, special cases, etc)
- Cut and partitioning problems (multicut, sparsest cut, multiway cut)
- Routing and flow problems (congestion minimization, unsplittable flow, disjoint paths, all-or-nothing flow)
- Bin packing (vector packing, geometric packing, connections to discrepancy)
- Discrepancy minimization
- Submodular optimization (maximization, minimization, constrained)
- SDP methods for constraint satisfaction problems.
- Sherali-Adams and SOS hierarchies in approximation.
- Hardness of approximation and inapproximability results
- Stochastic optimization with approximation guarantees
- Geometric approximation algorithms
- Tools and techniques in approximation (metric embeddings, optimization approaches, randomized techniques, nob-oblivious local search, rounding techniques, graph structure theory, etc)