COMP4801: Detailed Project Plan

Deterministic and Approximation Algorithms for Graph Theoretic Problems

Omer Wasim
Supervisor: Dr Hubert Chan

30th September, 2017

Contents

1 Introduction

The field of theoretical computer science is broad and vast. It includes a multitude of areas and draws from various disciplines-arguably most from mathematics. A few areas of active research includes the design and analysis of efficient algorithms in combinatorial optimization, algorithmic game theory, differential privacy, operations research and mechanism design. In general the field has grown vastly over the past few decades and strong results in theory lay the foundation and framework for applied research in numerous areas.

In this project, we will study only a small but greatly powerful selection of topics in combinatorial optimization for graph theoretic problems. Algorithms in this area normally come in 2 flavors-deterministic and non-deterministic(approximation/randomized) respectively. A deterministic algorithm is one which for any given input problem runs exactly the same way-the order of the steps executed by the algorithm are identical for any 2 given runs on certain underlying machine, while a non-deterministic algorithm is one for which the steps may be executed in possibly different orders and need not even be the same. A non-deterministic algorithm can be seen as one which makes certain choices during its execution so for 2 different runs on the same input, the steps taken and their order may not coincide. Consider now an algorithm for an optimization problem. Typically, the input includes an optimization objective, the underlying problem instance, and a set of constraints to be respected in a feasible solution, and an algorithm is evaluated on the basis of how well it meets the optimization objective. For a wide variety of useful problems in both theory and practice, there is no hope to find the optimal solution in polynomial time unless the $P = NP$ conjecture holds true. Thus, in these cases the goal of an algorithm designer is to design algorithms that find a feasible solution within some factor of the optimal solution.

More formally, an α-approximation algorithm is one which finds in polynomial time, a feasible solution whose cost is within an α factor of the optimal solution for *any* problem instance. Under the standard practice of defining an approximation ratio, it is easy to see that for a minimization objective $\alpha \geq 1$ and for a maximization objective, $\alpha \leq 1$. Current and past research has focused mainly on designing algorithms that have good approximation ratios and improving ones that have already been discovered. Recent work [5] on the Unique Games Conjecture has focused on the possibility of achieving certain approximation ratios for a various problems under the assumption that $P \neq NP$. Although the conjecture has not been proven yet, it is widely believed to hold and hence a major goal of researchers in this area is to close the gap between the current best ratio and that which is possible assuming the Unique Games conjecture holds true. Similar to showing that a certain problem is NP-complete via a reduction to a certain NP-complete problem already known, one can establish that no better approximation ratio is possible for a given problem via reduction from Unique Games.

Techniques that are normally used to design approximate algorithms include linear, convex and semidefinite programming [4] and using efficient randomization techniques. Low distortion metric embeddings have been used successfully in the past to approximate solutions to certain NP-hard problems. Let OPT denote the value of an optimal solution. Typically an approximation algorithm would include the following steps:

1) Formulate a linear/semidefinite program for a relaxed version of a problem. Show that the cost of this program P, denoted by $c(P)$ is bounded by the cost of the optimal solution, i.e. OPT.
2) Round the solution to obtain a feasible solution, say S.

3) Show that the cost of S, say $c(S)$ is within some factor of $c(P)$. Then it follows that $c(S)$ is within some factor of OPT.

Thus, approximation algorithms are quite useful in practice and banking on the assumption that linear/semidefinite programming can be solved efficiently, one obtains an approximation scheme in polynomial time under some compromise on the optimality of an application. This compromise has been shown in practice to be acceptable in many applications[7]. Note that the number of constraints of such a program could be exponential in input size, but via a good separation oracle, the number of constraints to be checked can be restricted to a polynomial number.

In the next section, we describe some problems which we will be focusing on in this project.

2 Background

Graph partitioning is a well studied problem in computer science. It has numerous practical applications-network design, image segmentation, task scheduling in multiprocessing environments and sparse gaussian elemination, to name a few. There are various settings of the graph partitioning problem. One which roughly captures the gist of the problem is: Given a graph (undirected) G, partition the graph into k pieces, such that each partition contains approximately the same number of vertices and the edge costs between those partitions are minimized. This is an NP-hard problem, therefore only approximation algorithms exist. Related problems include the balanced cut and its specific case for $k = 2$, the minimum bisection problem, both of which are NP-hard too.

A typical approach as described above is to balance the size of the partitions while minimizing the number (cost) of the cut edges between them. A recent framework of edge partitioning has been proposed, in which the number of edges are evenly distributed into k clusters while minimizing the cost objective which is commonly associated with minimizing the communication cost between vertices. The current best approximation ratio is $O(d_{max}\sqrt{log(n)log(k)})$ [2] where d_{max} is the maximum degree of any vertex. The natural open question to ask is: Is it possible to improve the approximation ratio-or remove the dependence on d_{max} completely. For the problem in which vertices are distributed to k different partitions whilst minimizing the cost of the edges between them, a $O(\sqrt{log(n)log(k)})$, where n is the number of vertices, is known[6]. What makes the edge partitioning problem difficult? Can we apply similar techniques [1],[6] to this problem too?

Clustering is a strongly related problem to graph partitioning and one which has been studied extensively under various objectives and settings. There has been recent work on hierarchical clustering,[3] and which uses SDP relaxations and metric embeddings, which might imply that similar techniques to related problems in graph partitioning might be applicable. In general, we aim to retain the focus of this project to these 2 broad problems, read seminal papers on both and investigate the open problems and settings that can be useful in the practical sense. The project will predominantly be focused on reading the vast amount of literature on these problems and more broadly on approximation algorithms including identifying potential areas in which approaches could be made better, theoretically. It may include evaluating certain algorithms experimentally as need be.

3 Objectives

Our key objectives are as follows:

1. A major objective of this thesis is to study and explore the techniques that have been used in the past to design efficient algorithms for approximation. Areas of focus will include low-distortion metric embeddings, linear programming and semidefinite programming. We will select papers from top conferences (typically in STOC, FOCS, SODA) and study those results.

2. We will identify the areas in which we can contribute to, such as improving the approximation ratio or the running times of algorithms. We would also investigate alternative and more efficient implementations of certain algorithms.

3. We will then work on a specific problem in a restricted setting that we will chose later and aim to improve on the merits as descrived above.

4. Currently, we do not expect to contribute towards an implementation, but instead try to bridge the theoretical gaps that exist. Although quite ambitious, we feel this is an integral element of our thesis.

5. Based on the outcome of the above, we might decide to instead work on an experimental evaluation. Towards this end, we would implement certain algorithms and compare their performances. Also, we will work on heuristics that may be relatively efficient in practice.

4 Goals and Methodology

Broadly speaking, the goal is to produce something original and of good quality-work which could potentially submitted to a conference/workshop proceeding. At a minimum we expect to meet the aforementioned objectives within the allotted time.

Since it is expected that the work will be mostly mathematical, discussions would be helpful in our case. Certain papers deemed to be useful will be read and fully understood. The open problems arising will be then discussed and investigated. Based on whether the theoretical improvement is possible within time constraints or otherwise due to the inherent difficulty of the problem, we might turn to an experimental evaluation and heuristic implementation.

The methodology will constantly evolve with the goals and may include programming at a late stage. Since at this point, our project is of a theoretical nature and significantly different than purely programming projects, we would appreciate if the difficulty of improving an approximation ratio is acknowledged and the time and effort put in to read and understand challenging mathematical and algorithmic results given due recognition.

5 Project Schedule

A rough time frame for the project is given as follows:

1. 1^{st} October to 15^{th} November: Read and discuss papers on graph partitioning/clustering that have appeared in top conferences from 2000-present. This would take significant time since the number of papers and the time to fully understand one takes at least 2-3 days (or more depending on the result).

2. 16^{th} November to 31^{st} December: Pursue some of the possible directions and try to improve/come up with a better algorithm for the problems in the papers read previously. Also identify any open problems of interest to the community and see if they could be feasibly tackled in the time allotted.

3. 1^{st} January to 15^{th} March: If there has been theoretical progress or if there is any reason to expect one, then continue working on the problem(s) and validate the results. Otherwise, try to come up with a heuristic implementation of the algorithms understood so far. The second portion of the final report, in the latter case, will include some implementation on real data.

4. 15^{th} March onward: Finalize the FYP report and present the project. Identify areas of further refinement.

6 References

[1] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. "Expander flows, geometric embeddings and graph partitioning". In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. Ed. by László Babai. ACM, 2004, pp. 222–231. ISBN: 1-58113-852-0. DOI: 10.1145/1007352.1007355. URL: http://doi.acm.org/10.1145/1007352.1007355.

[2] Florian Bourse, Marc Lelarge, and Milan Vojnovic. "Balanced Graph Edge Partition". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: ACM, 2014, pp. 1456–1465. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623660. URL: http://doi.acm.org/10.1145/2623330.2623660.

[3] Moses Charikar and Vaggos Chatziafratis. "Approximate Hierarchical Clustering via Sparsest Cut and Spreading Metrics". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. Ed. by Philip N. Klein. SIAM, 2017, pp. 841–854. ISBN: 978-1-61197-478-2. DOI: 10.1137/1.9781611974782.53. URL: https://doi.org/10.1137/1.9781611974782.53.

[4] Michel X. Goemans and David P. Williamson. "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming". In: *J. ACM* 42.6 (Nov. 1995), pp. 1115–1145. ISSN: 0004-5411. DOI: 10.1145/227683.227684. URL: http://doi.acm.org/10.1145/227683.227684.

[5] Subhash Khot. "On the Power of Unique 2-prover 1-round Games". In: *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*. STOC '02. Montreal, Quebec, Canada: ACM, 2002, pp. 767–775. ISBN: 1-58113-495-9. DOI: 10.1145/509907.510017. URL: http://doi.acm.org/10.1145/509907.510017.

[6] Robert Krauthgamer, Joseph (Seffi) Naor, and Roy Schwartz. "Partitioning Graphs into Balanced Components". In: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '09. New York, New York: Society for Industrial and Applied Mathematics, 2009, pp. 942–949. URL: http://dl.acm.org/citation.cfm?id=1496770.1496872.

[7] Vijay V. Vazirani. *Approximation Algorithms*. New York, NY, USA: Springer-Verlag New York, Inc., 2001. ISBN: 3-540-65367-8.