Unconstrained Submodular Maximization in MapReduce

Sikander Randhawa, Ben Chugg, Angad Kalra

The university of British Columbia
Vancouver, Canada

Background

Formally, a set function $f : 2^X \to \mathbb{R}$ is *submodular* iff

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B),$$

for all $A \subset B$.

Intuition: Think of decreasing marginal gains. The more you have, the less you appreciate gaining something new.

These functions find applications in graph theory, game theory and machine learning.

Background

Problem of interest:

$$\max_{S \subset X} f(S).$$

Has been well studied in many settings! In general this problem is NP-Hard [?].

So we settle for constraints and approximations:

- Greedy algorithm gives a $(1 - \frac{1}{e})$-approx. for Cardinality Constraints (monotone) [?]
- A continuous variant also gives a $(1 - \frac{1}{e})$-approx. for Matroid Constraints (monotone) [?].

Unconstrained, Non-Monotone Submodular Maximization (USM)

Less constraints $\Rightarrow$ harder to maximize. Our problem of interest:

- Unconstrained, non-monotone submodular function maximization,
- But, in a parallel setting (i.e. MapReduce)
  - Most applications of submodular maximization arise when dataset is too large to fit on one machine. Understanding how to parallelize computation is important.

MapReduce

What is MapReduce?

- Distributed framework for computation.
- Computation proceeds in rounds.
    - **At most polylog number of rounds permitted.**
- In each round, many machines perform computation.
    - Limited to polytime computation.
- After each round, machines communicate.
- Each machine limited to $o(n)$ storage.

This will be our distributed framework. Developed by Karloff et al. [**?**].

Our Approach

Explore canonical parallelizations of existing well-known algorithms.

One algorithm seemed better than others:

Local Search (Feige et al. [**?**]):

- Achieves $(\frac{1}{3})$ approximation factor.
  - Comparable to state of the art.
- There was an intuitive parallelization.
  - Split the search across many machines.
- Approximation guarantee would not deteriorate after parallelizing.
  - In fact, get approximation factor for free!

Local Search

Good algorithm for USM exists in the centralized setting!

Local Search (Idea)

1. Start with best singleton, $\{v\}$.
2. While we can improve the solution, repeat the following:
3. Add or remove an element as long as it increases our solution value.
4. Return the solution.

Number of total swaps is bounded by $\tilde{\mathcal{O}}(n^2) \Rightarrow$ halting guaranteed after $\tilde{\mathcal{O}}(n^2)$ rounds.

- Really need $\mathcal{O}(\log^k(n))$ rounds, so must widdle this down!

A Parallelization Approach

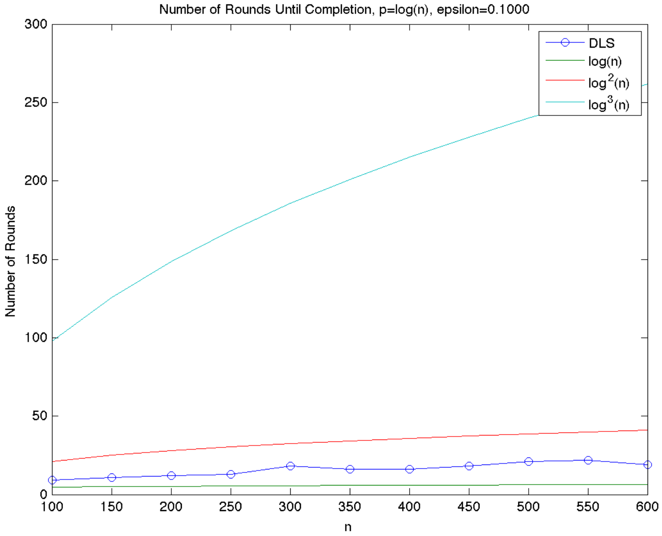
We have a promising algorithm, how do we parallelize?

- Start with arbitrary solution, S.
- Randomly split the universe, U into $\{U_i, \ldots, U_p\}$.
  - Machine i gets U_i.
- Each machine treats U_i as the full universe and performs local search starting from S.
- When all machines stop:
  - Update S to be the best solution across the machines.
- Repeat until S doesn't change.

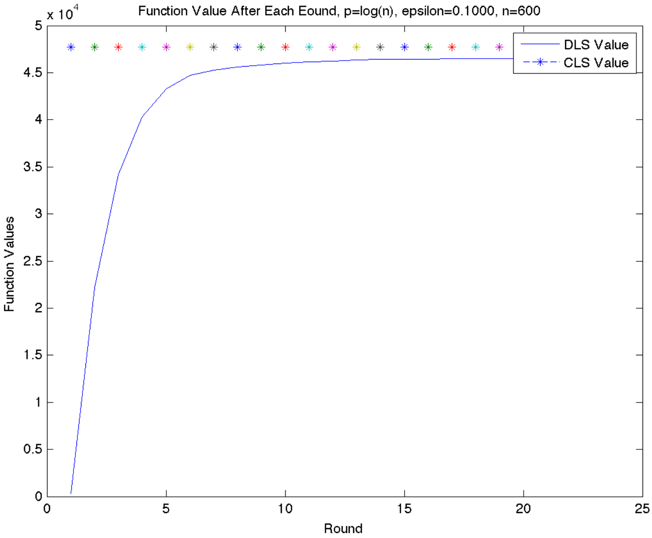
Main Problem: How do we guarantee that we only require a polylog number of rounds?

Turn to empirical evidence for building motivation, ideas, and intuition.

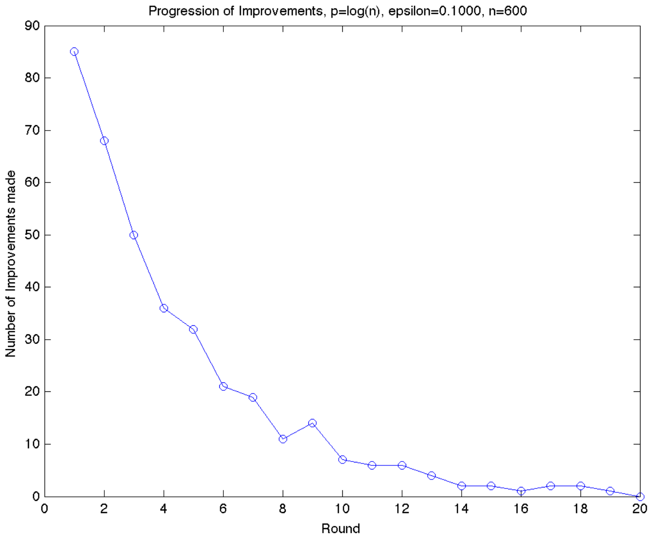
Number of Rounds



Function Value



Number of Improvements



References

Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz.
Submodular maximization with cardinality constraints.
Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1433–1452, 2014.

Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák.
Maximizing a submodular set function subject to a matroid constraint.
Integer Programming and Combinatorial Optimization: 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007. Proceedings, pages 182–196, 2007.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrak.
Maximizing non-monotone submodular functions.
SIAM Journal on Computing, 40.4:1133–1153, 2011.

Karloff Howard, Siddharth Suri ad, and Sergei Vassilvitskii.
A model of computation for mapreduce.
Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, pages 938–948, 2010.