CSC2420: Algorithm Design, Analysis and Theory Spring 2019

Allan Borodin

February 14, 2019

Week 6

Announcements:

- Assignment 1 is now complete and the due date February 25 which is the last date to drop a graduate course without penalty. Note: I will add what I think is an easier alternative to question 5 and awarding a bonus to anyone who solves both problems concerning matroids.
- Class next week or not?

Todays agenda

- Finish up discussion of oblivious local search $\frac{1}{2}$ approximation for maximizing a montone submodular function subject to a matroid constraint.
- Idea for obtaining a $1 - \frac{1}{e}$ approximation for maximizing a montone submodular function subject to an arbitrary matroid constraint. The weighted max coverage function.
- Introduction to randomized algorithms with applications to max-sat, unconstrained non-monotone submoodular maximization, online bipartite maximization.

Where we ended on February 7

The lecture on Thursday, Frebruary 7 basically ended with the statement of the standard greedy algorithm for maximizing a monotone submodular subject to a cardinality constraint. This algorithm provides a $1 - \frac{1}{e}$-approximation for a cardinality constraint. I added some additional slides for the purpose of the assignment. We will continue this week with a sketch of the proof of this approximation bound.

Generalizing to a matroid constraint

- Nemhauser Wolsey and Fisher [1978] showed that the $1 - \frac{1}{e}$ approximation is optimal in the sense that an exponential number of value oracle queries would be needed to beat the bound for the cardinalily constraint.
- Furthermore, Feige [1998] shows it is NP hard to beat this bound even for the explicitly represented maximum k-coverage problem.
- Following their first paper, Fisher, Nemhauser and Wolsey [1978] extended the cardinality constraint to a matroid constaint.
- Fisher, Nemhauser and Wolsey show that both the standard greedy algorithm and a 1-exchange local search algorithm (that will follow) achieve a $\frac{1}{2}$ approximation for maximzing a monotone submodular function subject to an arbitrary matroid constraint.
- They also showed that this bound was tight for the greedy and 1-exchange local search algorithms.

Monotone submodular maximization subject to a matroid constraint

We need some additional facts about matroids and submodular functions.

- Brualdi [1969] Let O and S be two independent sets in a matroid of the same size (in particular they could be two different bases of the same matroid). Then there is a bijection π between $O \setminus S$ and $S \setminus O$ such that for all $x \in O, (S \setminus \{\pi(x)\}) \cup x$ is independent.
- We have the following facts for a submodular function f on a ground set U:
  1. Let $C = \{c_1, \ldots, c_\ell\} \subseteq U \setminus S$. Then
  $$\sum_{i=1}^{\ell} [f(S + c_i) - f(S)] \geq f(S \cup C) - f(S)$$
  2. Let $\{t_1, \ldots, t_\ell\}$ be elements of S. Then
  $$\sum_{i=1}^{\ell} [f(S) - f(S \setminus \{t_i\}] \leq f(S)$$

The 1-exchange local search algorithm

We can start with any basis S (eg using the natural greedy algorithm). Then we keep trying to find an element of $x \notin S$ such that $(S \setminus \{\pi(x)\}) \cup \{x\} > f(S)$. Here π is the bijection as in Brualdi's result.

The previous local seach algorithm provides a $\frac{1}{2}$-approximation for maximizing a monotone submodular function.
Now let S be a local optimum and O an optimal solution. By local optimality, for all $x \in O \setminus S$, we have

$$f(S) \geq f((S \setminus \{\pi(x)\}) \cup \{x\})$$

Subtracting $(S \setminus \{\pi(x)\})$ from both sides, we have

$$f(S) - (S \setminus \{\pi(x)\}) \geq f((S \setminus \{\pi(x)\}) \cup \{x\}) - (S \setminus \{\pi(x)\})$$

From submodularity,

$$f((S \setminus \{\pi(x)\}) \cup \{x\}) - (S \setminus \{\pi(x)\}) \geq f(S \cup \{x\}) - f(S)$$

Thus for all $x \in O \setminus S$

$$f((S \setminus \{\pi(x)\} \geq f(S \cup \{x\}) - f(S)$$

Completing the local search approximation

Summing over all such x yields
$\sum_{x \in O \setminus S}[f(S) - f(S \setminus \{\pi(x)\})] \geq \sum_{x \in O \setminus S}[f(S \cup \{x\}) - f(S)]$
Applying the first fact on slide 28 to the right hand side of this inequality and the second fact to the left hand side, we get

$$f(S) \geq f(S \cup (O \setminus S)) - f(S) = f(O \cup S) - f(S) \geq f(O) - f(S)$$

which gives the desired $\frac{1}{2}$-approximation.

Achieving the $1 - \frac{1}{e}$ approximation for arbitrary matroids

- An open problem for 30 years was to see if the $1 - \frac{1}{e}$ approximation for the cardinality constraint could be obtained for arbitrary matroids.
- Calinsecu et al [2007, 2011] positively answer this open problem using a very different (than anything in our course) algorithm consisting of a continuous greedy algorithm phase followed by a pipage rounding phase.
- Following Calinsecu et al, Filmus and Ward [2012A, 2012B] develop (using LP analysis to guide their development) a sophisticated non-oblivious local search algorithm that is also able to match the $1 - \frac{1}{e}$ bound, first for the maximum coverage problem and then for arbitrary monotone submodular functions.

Another application of non-oblivious local search: weighted max coverage

The weighted max coverage problem

Given: A universe E, a weight function $w : E \rightarrow \Re^{\geq 0}$ and a collection of of subsets $\mathcal{F} = \{F_1, \ldots, F_n\}$ of E. The goal is to find a subset of indices S so as to maximize $f(S) = w(\cup_{i \in S} F_i)$ subject to some constraint (often a cardinality or matroid constraint). Note: f is a monotone submodular function.

- For $\ell < r = rank(M)$, the ℓ-flip oblivious local search for max coverage has locality gap $\frac{r-1}{2r-\ell-1} \rightarrow \frac{1}{2}$ as r increases. (Recall that greedy achieves $\frac{1}{2}$.)

The non-oblivious local search for max coverage

- Given two solutions S_1 and S_2 with the same value for the objective, we again ask (as we did for exact Max-k-Sat and the WMIS problem for a $k+1$ claw-free grpah), when is one solution better than the other?

The non-oblivious local search for max coverage

- Given two solutions S_1 and S_2 with the same value for the objective, we again ask (as we did for exact Max-k-Sat and the WMIS problem for a $k+1$ claw-free grpah), when is one solution better than the other?
- Similar to the motivation used in Max-k-Sat, solutions where various elements are covered by many sets is intuitively better so we are led to a potential function of the form $g(S) = \sum \alpha_{\kappa(u,S)} w(u)$ where $\kappa(u, S)$ is the number of sets F_i ($i \in S$) such that $u \in F_i$ and $\alpha : \{0, 1, \ldots, r\} \rightarrow \Re^{\geq 0}$.
- The interesting and non-trivial development is in defining the appropriate scaling functions $\{\alpha_i\}$ for $i = 0, 1, \ldots r$
- Filmus and Ward derive the following recurrence for the choice of the $\{\alpha_i\} : \alpha_0 = 0, \alpha_1 = 1 - \frac{1}{e}$, and $\alpha_{i+1} = (i+1)\alpha_i - i\alpha_{i-1} - \frac{1}{e}$.
- These α factors give more weight to those elements that appear frequently which makes it easier to swap out a set S and still keep many elements $u \in S$ in the collection.

The very high level idea and the locality gap

- The high-level idea behind the derivation is like the factor revealing LP idea used by Jain et al [2003]; namely, Filmus and Ward formulate an LP for an instance of rank r that determines the best obtainable ratio (by this approach) and the $\{\alpha_i\}$ obtaining this ratio.

The Filmus-Ward locality gap for the non oblivious local search

The 1-flip non oblivious local search has locality gap $O(1 - \frac{1}{e} - \epsilon)$ and runs in time $O(\epsilon^{-1} r^2 |\mathcal{F}||U| \log r)$
The ϵ in the ratio can be removed using partial enumeration resulting in time $O(r^3 |\mathcal{F}|^2 |U|^2 \log r)$.

A non oblivious local search for an arbitrary monotone submodular function

- The previous development and the analysis needed to obtain the bounds is technically involved but is aided by having the explicit weight values for each F_i. For a general monotone submodular function we no longer have these weights.
- Instead, Filmus and Ward define a potential function g that gives extra weight to solutions that contain a large number of good sub-solutions, or equivalently, remain good solutions on average even when elements are randomly removed.
- A weight is given to the average value of all solutions obtained from a solution S by deleting i elements and this corresponds roughly to the extra weight given to elements covered $i+1$ times in the max coverage case.
- Letting $\beta_k^{(|S|)}$ be the weight given to the average value of $f(T)$ over all subsets T of size k, the potential function is :

  $g(S) = \sum_{k=0}^{|S|} \sum_{T:T\subseteq S,|T|=k} \frac{\beta_k^{(|S|)}}{\binom{|S|}{k}} f(T) = \sum_{k=0}^{|S|} \beta_k^{(|S|)} \mathbf{E}_T[f(T)]$

Randomized algorithms

Our next theme will be randomized algorithms. For the main part, our previous themes have been on algorithmic paradigms, so far variants of greedy and local-search.. Randomization is not per se an algorithmic paradigm (in the same sense as greedy algorithms, DP, local search, LP rounding, primal dual algorithms).

Randomized algorithms

Our next theme will be randomized algorithms. For the main part, our previous themes have been on algorithmic paradigms, so far variants of greedy and local-search.. Randomization is not per se an algorithmic paradigm (in the same sense as greedy algorithms, DP, local search, LP rounding, primal dual algorithms).

Rather, randomization can be thought of as an additional algorithmic idea that can be used in conjuction with any algorithmic paradigm. However, its use is so prominent and varied in algorithm design and analysis, that it takes on the sense of an algorithmic way of thinking.

The why of randomized algorithms

- There are some problem settings (e.g. simulation, cryptography, interactive proofs, sublinear time algorithms) where randomization is *necessary*.
- We can use randomization to improve approximation ratios.
- Even when a given algorithm can be efficiently derandomized, there is often conceptual insight to be gained from the initial randomized algorithm.
- In complexity theory a fundamental question is how much can randomization lower the time complexity of a problem. For decision problems, there are three polynomial time randomized classes ZPP (zero-sided), RP (1-sided) and BPP (2-sided) error. The big question (and conjecture?) is BPP $=$ P?
- One important aspect of randomized algorithms (in an offline setting) is that the probability of success can be amplified by repeated independent trials of the algorithm.

Some applications of randomized algorithms to the online setting

In addition to the important role of randomiztion in the more standard offline algorithm setting, randomization plays a very central role in online algorithms as the online setting is particularly vulnerable to worst case adversarial examples. Here are some problems we will consider in the online setting:

1. Naive exact max-k-sat algorithm
2. De-randomization by the method of conditional expectation
3. The KVV algorithm for online unweighted bipartite matching
4. The Buchbinder et al two sided online greedy algorithm for the unconstrined maximization of a non-monotone submodular function. and application to max-sat.
5. Online with advice and relation to randomized online algorithms
6. De-randomization using two and multi pass algorithms

In addition we will consider some "classical" online problems which continue to be studied.

But first a few more comments on randomization and complexty theory.

Some problems in randomized polynomial time not known to be in polynomial time

1. The symbolic determinant problem.
2. Given n, find a prime in $[2^n, 2^{n+1}]$
3. Estimating volume of a convex body given by a set of linear inequalitiies.
4. Solving a quadratic equation in $Z_p[x]$ for a large prime p.

We will see that often a naive randomization provides the best current results. One can think of *naive randomization* as a paradigm. That is, instead of looking for a particular solution, try a random solution.

Polynomial identity testing

- The general problem concerning polynomial identities is that we are implicitly given two multivariate polynomials and wish to determine if they are identical. One way we could be implicitly given these polynomials is by an arithmetic circuit. A specific case of interest is the following symbolic determinant problem.
- Consider an $n \times n$ matrix $A = (a_{i,j})$ whose entries are polynomials of total degree (at most) d in m variables, say with integer coeficients. The determinant $det(A) = \sum_{\pi \in S_n} (-1)^{sgn(\pi)} \prod_{i=1}^{n} a_{i,\pi(i)}$, is a polynomial of degree nd. The symbolic determinant problem is to determine whether $det(A) \equiv \mathbf{0}$, the zero polynomial.

Schwartz-Zipple Lemma

Schwartz Zipple Lemma

Let $P \in \mathbf{F}[x_1, \ldots, x_m]$ be a non zero polynomial over a field $\mathbf{F}$ of total degree at most d. Let S be a finite subset of $\mathbf{F}$. Then $Prob_{r_i \in_u S}[P(r_1, \ldots . r_m) = 0] \leq \frac{d}{|S|}$

Schwartz Zipple is clearly a multivariate generalization of the fact that a univariate polynomial of degree d can have at most d zeros.

Polynomial identity testing and symbolic determinant continued

- Returning to the symbolic determinant problem, suppose then we choose a suffciently large set of integers S (for definiteness say $|S| \geq 2nd$). Randomly choosing $r_i \in S$, we evaluate each of the polynomial entries at the values $x_i = r_i$. We then have a matrix A' with (not so large) integer entries.
- We know how to compute the determinant of any such integer matrix $A'_{n \times n}$ in $O(n^3)$ arithmetic operations. (Using the currently fastest, but not necessarily practical, matrix multiplication algorithm, the determinant can be computed in $O(n^{2.373})$ arithmetic operations.)
- That is, we are computing the $det(A)$ at random $r_i \in S$ which is a degree nd polynomial. Since $|S| \geq 2nd$, then $Prob[det(A') = 0] \leq \frac{1}{2}$ assuming $det(A) \not\equiv \mathbf{0}$. The probability of correctness con be amplifed by choosing a bigger S or by repeated trials.
- In complexity theory terms, the problem (is $det(A) \equiv \mathbf{0}$) is in co-RP.

The naive randomized algorithm for exact Max-k-Sat

We continue our discussion of randomized algorithms by considering the use of randomization for improving approximation algorithms. In this context, randomization can be (and is) combined with any type of algorithm.
Note: For the following maximization problems, we will follow the prevailing convention by stating competitive ratios as fractions $c < 1$.

- Consider the exact Max-k-Sat problem where we are given a CNF propositional formula in which every clause has exactly k literals. We consider the weighted case in which clauses have weights. The goal is to find a satisfying assignment that maximizes the size (or weight) of clauses that are satisfied.
- As already noted, since exact Max-k-Sat generalizes the exact k- SAT decision problem, it is clearly an NP hard problem for $k \geq 3$. It is interesting to note that while 2-SAT is polynomial time computable, Max-2-Sat is still NP hard.
- The naive randomized (online) algorithm for Max-k-Sat is to randomly set each variable to *true* or *false* with equal probability.

Analysis of naive Max-k-Sat algorithm continued

- Since the expectation of a sum is the sum of the expectations, we just have to consider the probability that a clause is satisfied to determine the expected weight of a clause.
- Since each clause C_i has k variables, the probability that a random assignment of the literals in C_i will set the clause to be satisfied is exactly $\frac{2^k-1}{2^k}$. Hence $\mathbf{E}$ [weight of satisfied clauses] $= \frac{2^k-1}{2^k}\sum_i w_i$
- Of course, this probability only improves if some clauses have more than k literals. It is the small clauses that are the limiting factor in this analysis.
- This is not only an approxination ratio but moreover a "totality ratio" in that the algorithms expected value is a factor $\frac{2^k-1}{2^k}$ of the sum of all clause weights whether satisfied or not.
- We can hope that when measuring against an optimal solution (and not the sum of all clause weights), small clauses might not be as problematic as they are in the above analysis of the naive algorithm.

Derandomizing the naive algorithm

We can derandomize the naive algorithm by what is called the method of conditional expectations. Let $F[x_1,\ldots,x_n]$ be an exact k CNF formula over n propositional variables $\{x_i\}$. For notational simplicity let *true* $= 1$ and *false* $= 0$ and let $w(F)|\tau$ denote the weighted sum of satisfied clauses given truth assignment τ.

- Let x_j be any variable. We express $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}}]$ as $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}}|x_j = 1] \cdot (1/2) + \mathbf{E}[w(F)|_{x_i \in_u \{0,1\}}|x_j = 0] \cdot (1/2)$
- This implies that one of the choices for x_j will yield an expectation at least as large as the overall expectation.
- It is easy to determine how to set x_j since we can calculate the expectation clause by clause.
- We can continue to do this for each variable and thus obtain a deterministic solution whose weight is at least the overall expected value of the naive randomized algorithm.
- NOTE: The derandomization can be done so as to achieve an online algorithm. Here the (online) input items are the propostional variables. What input representation is needed/sufficient?

(Exact) Max-k-Sat

- For exact Max-2-Sat (resp. exact Max-3-Sat), the approximation (and totality) ratio is $\frac{3}{4}$ (resp. $\frac{7}{8}$).
- For $k \geq 3$, using PCPs (probabilistically checkable proofs), Hastad proves that it is NP-hard to improve upon the $\frac{2^k-1}{2^k}$ approximation ratio for Max-k-Sat.
- For Max-2-Sat, the $\frac{3}{4}$ ratio can be improved by the use of semi-definite programming (SDP) and randomized rounding.
- The analysis for exact Max-k-Sat clearly needed the fact that all clauses have at least k clauses. What bound does the naive online randomized algorithm or its derandomztion obtain for (not exact) Max-2-Sat or arbitrary Max-Sat (when there can be unit clauses)?

Johnson's Max-Sat Algorithm

Johnson's [1974] algorithm

For all clauses C_i, $w'_i := w_i/(2^{|C_i|})$

Let L be the set of clauses in formula F and X the set of variables

For $x \in X$ (or until L empty)

  Let $P = \{C_i \in L$ such that x occurs positively$\}$

  Let $N = \{C_j \in L$ such that x occurs negatively$\}$

  If $\sum_{C_i \in P} w'_i \geq \sum_{C_j \in N} w'_j$

    $x := true; L := L \setminus P$

    For all $C_r \in N$, $w'_r := 2w'_r$ **End For**

  Else

    $x := false; L := L \setminus N$

    For all $C_r \in P$, $w'_r := 2w'_r$ **End For**

  End If

  Delete x from X

End For

Aside: This reminds me of boosting (Freund and Shapire [1997])

Johnson's algorithm is the derandomized algorithm

- Twenty years after Johnson's algorithm, Yannakakis [1994] presented the naive algorithm and showed that Johnson's algorithm is the derandomized naive algorithm.
- Yannakakis also observed that for arbitrary Max-Sat, the approximation of Johnson's algorithm is at best $\frac{2}{3}$. For example, consider the 2-CNF $F = (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge \bar{y}$ when variable x is first set to true. Otherwise use $F = (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge y$.
- Chen, Friesen, Zheng [1999] showed that Johnson's algorithm achieves approximation ratio $\frac{2}{3}$ for arbitrary weighted Max-Sat.
- For arbitrary Max-Sat (resp. Max-2-Sat), the current best approximation ratio is .7968 (resp. .9401) using semi-definite programming and randomized rounding.

  Note: While existing combinatorial algorithms do not come close to these best known ratios, it is still interesting to understand simple and even online algorithms for Max-Sat.

Modifying Johnson's algorithm for Max-Sat

- In proving the (2/3) approximation ratio for Johnson's Max-Sat algorithm, Chen et al asked whether or not the ratio could be improved by using a random ordering of the propositional variables (i.e. the input items). This is an example of the random order model (ROM), a randomized variant of online algorithms.
- To precisely model the Max-Sat problem within an online or priority framework, we need to specify the input model.
- In increasing order of providing more information (and possibly better approximation ratios), the following input models can be considered:

Model 0 Each propositional variable x is represented by the names of the positive and negative clauses in which it appears.

Model 1 Each propositional variable x is represented by the length of each clause C_i in which x appears positively, and for each clause C_j in which it appears negatively.

Model 2 In addition, for each C_i and C_j, a list of the other variables in that clause is specified.

Model 3 The variable x is represented by a complete specification of each clause it which it appears.

Improving on Johnson's algorithm

- The question asked by Chen et al was answered by Costello, Shapira and Tetali [2011] who showed that in the ROM model, Johnson's algorithm achieves approximation $(2/3 + \epsilon)$ for $\epsilon \approx .003653$
- Poloczek and Schnitger [same SODA 2011 conference] show that the approximation ratio for Johnsons algorithm in the ROM model is at most $2\sqrt{157} \approx .746 < 3/4$, noting that $\frac{3}{4}$ is the ratio first obtained by Yannakakis' IP/LP approximation that we will soon present.
- Poloczek and Schnitger first consider a "canonical randomization" of Johnson's algorithm; namely, the canonical randomization sets a variable $x_i = \mathit{true}$ with probability $\frac{w'_i(P}{w'_i(P)+w'_i(N)}$ where $w'_i(P)$ (resp. $w'_i(N)$) is the current combined weight of clauses in which x_i occurs positively (resp. negatively). Their substantial additional idea is to adjust the random setting so as to better account for the weight of unit clauses in which a variable occurs.

A few comments on the Poloczek and Schnitger algorithm

- The Poloczek and Schnitger algorithm is called Slack and has approximation ratio $= 3/4$.
- The Slack algorithm is a randomized online algorithm (i.e. adversary chooses the ordering) where the variables are represented within input Model 1.
- This approximation ratio is in contrast to Azar et al [2011] who prove that no randomized online algorithm can achieve approximation better than $2/3$ when the input model is input model 0.
- Finally (in this regard), Poloczek [2011] shows that no deterministic priority algorithm can achieve a $3/4$ approximation within input model 2. This provides a sense in which to claim the that Poloczek and Schnitger Slack algorithm "cannot be derandomized".
- The best deterministic priority algorithm in the third (most powerful) model remains an open problem as does the best randomized priority algorithm and the best ROM algorithm.

Revisiting the "cannot be derandomized comment"

Spoiler alert: we will be discussing how algorithms that cannot be derandomized in one sense can be deramdomized in another sense.

- The Buchbinder et al [2012] online randomized 1/2 approximation algorithm for Unconstrained Submodular Maximization (USM) cannot be derandomized into a "similar" deterministic algorithm by a result of Huang and Borodin [2014].
- However, Buchbinder and Feldman [2016] show how to derandomize the Buchbinder et al algorithm into an algorithm that generates $2n$ parallel streams where each stream is an online algorithn.
- The Buchbinder et al USM algorithm is the basis for a randomized 3/4 approximation online MaxSat (even Submodular Max Sat) algorithm.
- Pena and Borodin show how to derandomize this 3/4 approximation algorithm following the approach of Buchbinder and Feldman.
- Poloczek et al [2017] de-randomize an equivalent Max-Sat algorithm using a 2-pass online algorithm.