CSC2420 Spring 2016: Algorithm Design, Analysis and Theory
Lecture 11

Allan Borodin

March 31, 2016

Announcements and todays agenda

- Announcements
  1. With much embarrassment, I still have three sets of slides for last weeks guest lecture (Lecture 8) by Aleksander Nikolov that I still haven't had a chance to read carefully. I really really hope to have them posted by this weekend.
  2. There are now 4 questions posted for Assignment 3.
- Todays agenda
  1. Discussion of randomized primality testing.
  2. Discussion of weighted majority algorithm
  3. Monotone submodular maximization subject to matroid and independence constraints and the return of non-oblivious local search.
  4. The Lovasz Local lemma and the Moser-Tardos algorithm for finding a satisfying instance of an exact k-SAT formula in which every clause C shares a variable with at most $d < 2^k/e$ other clauses.

Primality testing

- I now want to briefly turn attention to one of the most influential randomized algorithms, namely a poly time randomized algorithm for primality (or perhaps better called compositeness) testing. Let $PRIME = \{N | N \text{ is a prime number}\}$ where N is represented in say binary (or any base other than unary) so that $n = |N| = O(\log N)$.
- History of polynomial time algorithms:
  1. Vaughan 1972 showed that *PRIMES* is in *NP*. Note that co-PRIMES (i.e. the composites) are easily seen to be in *NP*.
  2. One sided error randomized algorithms (for compositeness) by Solovay and Strassen and independently Rabin in 1974. That is, $Prob[ALG \text{ says } N \text{ prime } | N \text{ composite}] \leq \delta < 1$ and $Prob[ALG \text{ says } N \text{ composite } | N \text{ prime}] = 0$
  3. The Rabin test is related to an algorithm by Miller that gives a deterministic polynomial time algorithm assuming a conjecture that would follow from (the unproven) ERH. The Rabin test is now called the Miller-Rabin test.
  4. Goldwasser and Killian establish a 0-sided randomized algorithm.
  5. In 2002, Agarwal, Kayal and Saxena show that primality is in deterministic polynomial time.

Why consider randomized tests when there is a deterministic algorithm?

- Even though there is now a deterministic algorithm, it is not nearly as efficient as the 1-sided error algorithms which are used in practice. These randomized results spurred interest in the topic (and other number theoretic algorithms) and had a major role in cryptographic protocols (which often need random large primes). Moreover, these algorithms became the impetus for major developments in randomized algoritms.
- While many of our previous algorithms (excluding the streaming algorithm for F_k) might be considered reasonably natural (or natural extensions of a deterministic algorithm), the primality tests require some understanding of the subject matter (i.e. a little number theory) and these algorithms are not something that immediately comes to mind.

Some basic number theory we need

- $Z_N^* = \{a \in Z_N : gcd(a, N) = 1\}$ is a (commutative) group under multiplication mod N.
- If N is prime, then
  1. For $a \neq 0(modN), a^{N-1} = 1(modN)$.
  2. Z_N^* is a cyclic group; that is there exists a generator g such that $\{g, g^2, g^3, \ldots, g^{N-1}\}$ (all mod N) is the set Z_N^*. This implies that $g^i \neq 1(modN)$ for any $1 \leq i < N - 1$.
  3. There are exactly two square roots of 1 in Z_N^*, namely 1 and -1.
- The Chinese Remainder Theorem: Whenever N_1 and N_2 are relatively prime (i.e. $gcd(N_1, N_2) = 1$), then for all $v_1 < N_1$ and $v_2 < N_2$, there exists a unique $w < N_1 \cdot N_2$ such that $v_1 = w(modN_1)$ and $v_2 = w(modN_2)$.

A simple but "not quite" correct algorithm

We also need two basic computational facts.

1. a^i mod N can be computed efficiently.
2. $gcd(a, b)$ can be efficiently computed.

The following is a simple algorithm that works except for an annoying set of numbers called Carmichael numbers.

Simple algorithm ignoring Carmichael numbers

Choose $a \in Z_N$ uniformly at random.
If $gcd(a, N) \neq 1$, then Output Composite
If a^{N-1} mod $N \neq 1$, then Output Composite
Else Output Prime

When does the simple algorithm work?

- $S = \{a | gcd(a, N) = 1 \text{ and } a^{N-1} = 1\}$ is a subgroup of Z_N^*
- If there exists an $a \in Z_N^*$ such that $gcd(a, N) = 1$ but $a^{N-1} \neq 1$, then S is a proper subgroup of Z_N^*.
- By Lagrange's theorem, if S is a proper subgroup, $|S|$ must divide the order of the group so that $|S| \leq \frac{N-1}{2}$
- Thus the simple algorithm would be a 1-sided error algorithm with probabiltiy $< \frac{1}{2}$ of saying Prime when N is Composite.
- The only composite numbers that give us trouble are the Carmichael numbers (also known as *false primes*) for which $a^{N-1} mod N = 1$ for all a such that $gcd(a, N) = 1$.
- It was only recently (relatively speaking) that in 1994 it was proven that there are an infinite number of Carmichael numbers.
- The first three Carmicahel numbers are 561, 1105, 1729

Miller-Rabin 1-sided error algorithm

Let $N-1=2^t u$ with u odd %Since wlg. N is odd, $t \geq 1$
Randomly choose non zero $a \in Z_N$ %Hoping that a will be composite certificate
If $gcd(a, N) \neq 1$ then report Composite
$x_0 = a^u$ %All computation is done mod N
For $i = 1 \ldots t$
    $x_i := x_{i-1}^2$
    If $x_i = 1$ and $x^{i-1} \notin \{-1, 1\}$, then report Composite
End For
If $x_t \neq 1$, then report Composite $\%x_t = x^{N-1}$
Else report Prime

Analysis sketch of Miller-Rabin

- Let S be the set of $a \in N$ that pass (i.e. fool) the Rabin-Miller test.
- S is a subgroup of Z_N^*. We want to show that S is a proper subgroup and then as before by Langrange we will be done.
- It suffices then to find one element $w \in Z_N^*$ that will not pass the Miller-Rabin test.
  Case 1: N is not Carmichael and then we are done.
  Case 2: N is Carmichael and hence N cannot be a prime power.
  - $N = N_1 \cdot N_2$ and $gcd(N_1, N_2) = 1$ and of course odd
  - The non-certificates must include some b such that $b^{2^i u} = -1(modN)$ and hence $b^{2^i u} = -1(modN_1)$
  - By the Chinese Remainder Theorem, there exists $w = v(modN_1)$ and $w = 1(modN_2)$
  - Hence $w^{2^i u} = -1(modN_1)$ and $w^{2^i u} = 1(modN_2)$
  - This implies $w^{2^i u} \notin \{-1, 1\} \pmod N$

New topic: the weighted majority algorithm

I am following a survey type paper by Arora, Hazan and Kale [2008]. To quote from their paper: "We feel that this meta-algorithm and its analysis should be viewed as a basic tool taught to all algorithms students together with divide-and-conquer, dynamic programming, random sampling, and the like".

- The weighted majority algorithm and generalizations
  The "classical" WMA pertains to the following situation:
  Suppose we have say n expert weathermen (or maybe "expert" stock market forecasters) and at every time t, they give a binary prediction (rain or no rain, Raptors win or lose, dow jones up or down, Canadian dollar goes up or down.
- Now some or all of these experts may actually be getting their opinions from the same sources (or each other) and hence these predictions can be highly correlated.
- Without any knowledge of the subject matter (and why should I be any different from the "experts") I want to try to make predictions that will be nearly as good (over time t) as the BEST expert.

The weighted majority algorithm

The WM algorithm

Set $w_i(0) = 1$ for all i

For $t = 0...$

Our $(t+1)^{st}$ predication is

0: if $\sum_{\{i:\text{ expert } i \text{ predicts } 0\}} w_i(t) \geq (1/2)\sum_i w_i(t)$

1: if $\sum_{\{i:\text{ expert } i \text{ predicts } 1\}} w_i(t) \geq (1/2)\sum_i w_i(t)$; arbitrary o.w.

% We vote with weighted majority; arbitrary if tie

For $i = 1..n$

If expert i made a mistake on $(t+1)^{st}$ prediction

then $w_i(t+1) = (1-\epsilon)w_i(t)$;

else $w_i(t+1) = w_i(t)$

End If

End For

End For

How good is our uninformed MW prediction?

Theorem : Perfomance of WM

Theorem: Let $m_i(t)$ be the number of mistakes of expert i after the first t forecasts, and let $M(t)$ be the number of our mistakes. Then for any expert i (including the best expert) $M(t) \leq \frac{2 \ln n}{\epsilon} + 2(1+\epsilon)m_i(t)$.

- That is, we are "essentially" within a multiplicative factor of 2 plus an additive term of the best expert (without knowing anything).
- Using randomization, the factor of 2 can be removed. That is, instead of taking the weighted majority opinion, in each iteration t, choose the prediction of the i^{th} expert with probability $w_i(t)/\sum_i w_i(t)$

Theorem: Performance of Randomized WM

For any expert i, $\mathbf{E}[M(t)] \leq \frac{\ln n}{\epsilon} + (1+\epsilon)m_i(t)$

Proof of deterministic WM

Let's assume that $\epsilon \leq 1/2$. It follows that

$$-\epsilon - \epsilon^2 \leq \ln(1-\epsilon) < -\epsilon$$

Let $w_{i,t}$ be the weight of the i^{th} expert at time t and let $m_i(t)$ be the number of mistakes made by expert i. . Consider the potential function $\Phi(t) = \sum_i w_{i,t}$. Clearly

$$\Phi(t) \geq w_{i,t} = (1-\epsilon)^{m_i(t)}$$

We now need an upper bound on $\Phi(t)$. Since each time the WM algoriithm makes a mistake, at least half of the algorithms make a mistake so that $\Phi(t) \leq (1-\epsilon/2)\Phi(t-1)$. Starting with $\Phi(0) = n$, by induction

$$\Phi(t) \leq n \cdot (1-\epsilon/2)^{M(t)}$$

Putting the two inequlaities together and taking logarithms

$$\ln(1-\epsilon)m_i(t) \leq \ln n + M(t)\ln(1-\epsilon/2)$$

The argument is completed by rearranging, using the above facts concerning $\ln(1-\epsilon)$ and then dividing by $\epsilon/2$.

What is the meaning of the randomized impovement?

- In many applications of randomization we can argue that randomization is (provably) necessary and in other applications, it may not be provable so far but current experience argues that the best algorithm in theory and practice is randomized.
- For some algorithms (and especially online algorithms) analyzed in terms of worst case performance, there is some debate on what randomization is actually accomplishing.
- In a [1996] article Blum states that "Intuitively, the advantage of the randomized approach is that it dilutes the worst case". He continues to explain that in the determinstic algorithm, slightly more than half of the total weight could have predicted incorrectly, causing the algorithm to make a mistake and yet only reducing the total weight by $1/4$ (when $\epsilon = 1/2$). But in the randomized version, there is still a .5 probability that the algorithm will predict correctly. Convincing?

An opposing viewpoint

- In the blog LessWrong this view is strongly rejected. Here the writer makes the following comments: "We should be especially suspicious that the randomized algorithm guesses with probability proportional to the expert weight assigned. This seems strongly reminiscent of betting with 70% probability on blue, when the environment is a random mix of 70% blue and 30% red cards. We know the best bet and yet we only sometimes make this best bet, at other times betting on a condition we believe to be less probable.
  Yet we thereby prove a smaller upper bound on the expected error. Is there an algebraic error in the second proof? Are we extracting useful work from a noise source? Is our knowledge harming us so much that we can do better through ignorance?" The writer asks: "So what's the *gotcha* ... the improved upper bound proven for the randomized algorithm did not come from the randomized algorithm making systematically better predictions - doing superior cognitive work, being more intelligent - but because we arbitrarily declared that an intelligent adversary could read our mind in one case but not in the other."

Further defense of the randomized approach

- Blum's article expresses a second benefit of the randomized approach: "Therefore the algorithm can be naturally applied when predictions are 'strategies' or other sorts of things that cannot easily be combined together. Moreover, if the 'experts' are programs to be run or functions to be evaluated, then this view speeds up prediction since only one expert needs to be examined in order to produce the algorithm's prediction ...."
- We also know (in another context) that ROM ordering can beat any deterministic priority order say for the online bipartite matching problem.

Generalizing: The Multiplicative Weights algorithm

The Weighted Majority algorithm can be generalized to the multiplicative weights algorithm. If the i^{th} expert or decision is chosen on day t, it incurs a real valued cost/profit $m_i(t) \in [-1, 1]$. The algorithm then updates $w_i(t+1) = (1 - \epsilon m_i(t))w_i(t)$. Let $\epsilon \leq 1/2$ and $\mathbf{\Phi}(t) = \sum_i w_i(t)$. On day t, we randomly select expert i with probability $w_i(t)/\Phi(t)$.

Performance of The MW algorithm

The expected cost of the MW algorithm after T rounds is $\sum_{t=1}^{T} \mathbf{m}(t) \cdot \mathbf{p}(t) \leq \frac{\ln n}{\epsilon} + \sum_{t=1}^{T} m_i(t) + \epsilon \sum_{t=1}^{T} |m_i(t)|$

Reinterpreting in terms of gains instead of losses

We can have a vector $\mathbf{m}(t)$ of gains instead of losses and then use the "cost vector" $-\mathbf{m}(t)$ in the MW algorithm resulting in:

Performance of The MW algorithm for gains

$\sum_{t=1}^{T} \mathbf{m}(t) \cdot \mathbf{p}(t) \geq -\frac{\ln n}{\epsilon} + \sum_{t=1}^{T} m_i(t) - \epsilon \sum_{t=1}^{T} |m_i(t)|$

By taking convex combinations, an immediate corollary is

Performance wrt. a fixed distribution p

$\sum_{t=1}^{T} \mathbf{m}(t) \cdot \mathbf{p}(t) \geq -\frac{\ln n}{\epsilon} + \sum_{t=1}^{T} \mathbf{m}(t) - \epsilon |\mathbf{m}(t)|)\mathbf{p}$

An application to learning a linear binary classifier

Instead of the online application of following expert advice, let us now think of "time" as rounds in an iterative procedure. In particular, we would like to compute a linear binary classifier (when it exists).

- We are trying to classsify objects characterized by n features; that is by points $\mathbf{a}$ in $\Re^n$. We are given m labelled examples $(\mathbf{a}_1, \ell_1), \ldots, (\mathbf{a}_m, \ell_m)$ where $\ell_j \in \{-1, +1\}$
- We are going to assume that these examples can be "well classified" by a linear classifier in the sense that there exists a non negative vector $\mathbf{x}^* \in \Re^n$ (with $x_i \geq 0$) such that $sign(\mathbf{a}_j \cdot \mathbf{x}^*) = \ell_j$ for all j.
- This is equivalent to saying $\ell_j \mathbf{a}_j \cdot \mathbf{x}^* \geq 0$ and furthermore (to explain the "well") we will say that $\ell_j \mathbf{a}_j \cdot \mathbf{x}^* \geq \delta$ for some $\delta > 0$.
- The goal now is to learn some linear classifer; ie a non negative $\mathbf{x} \in \Re^n$ such that $\ell_j \mathbf{a}_j \cdot \mathbf{x}^* \geq 0$. Without loss of generality, we can assume that $\sum_i x_i = 1$.
- Letting $\mathbf{b}_j = \ell_j \mathbf{a}_j$, this can now be veiwed as a reasonably general LP (search) problem.

Littlestone's Winnow algorithm for learning a linear classifier

- Litlestone [1987] used the multiplicative weights approach to solve this linear classification problem.
- Let $\rho = \max_j ||\mathbf{b}_j||_\infty$ and let $\epsilon = \delta/(2\rho)$
- The idea is to run the MW algorthm with the decisions given by the n features and gains specified by the m examples. The gain for feature i with respect to the j^{th} example is defined as $(\mathbf{b}_j)_i/\rho$ which is in [-1,1]. The $\mathbf{x}$ we are seeking is the distribution $\mathbf{p}$ in MW.

The Winnow algorithm

Initialize $\mathbf{p}$
While there are points not yet satisfied
  Let $\mathbf{b}_j \cdot \mathbf{p} < 0$ % a constraint not satisfied
  Use MW to upate $\mathbf{p}$
End While

Bound on number of iterations

The Winnow algorithm will terminate in at most $\lceil 4\rho^2 \ln n/\delta^2 \rceil$ iterations.

Some additional remarks on Multiplicative Weights

The survey by Arora, Hazan and Kale [2012] discusses other modifications of the MW paradigm and numerous applications. In terms of applications, they sketch results for

- Aporoximately solving (in the sense of property testing) the decision problem for an LP; there that is given linear constraints expressed by $A\mathbf{x} \geq \mathbf{b}$, the decision problem is to see if such a non-negative $\mathbf{x}$ exists (or more generally, if $\mathbf{x}$ is in some given convex set). The algorithm either returns a $\mathbf{x} : \mathbf{A_i x} \geq \mathbf{b_i} - \delta$ for all i and some additive approximation δ or says that the given LP was infeasible.
- Solving zero sum games approximately.
- The AdaBoost algorithm of Shapire and Freund
- Some other specific applications including a class of online algorithms.

Matroids, k-independence systems, submodular functions and *the* natural greedy algorithm.

- Beautiful development starting in the 1950's with the work of Rado [1957], Gale 1968 and Edmonds [1970, 1971], (extended by Korte and Lovász [1981, 1984], and others) as to contexts in which *"the natural"* greedy algorithm will produce an optimal solution.
- In particular, matroids characterize those hereditary set systems for which the *natural* greedy algorithm (determined by the order $c_1 \geq c_2 \ldots$ for maximization) will optimize a linear objective function $\sum c_i x_i$ for a maximum independent set $\{i : x_i = 1\}$ in a matroid $M = (E, \mathcal{I})$ where $\mathcal{I}$ are the independent subsets of E.
- Here the best known example is perhaps the minimum (or maximum) spanning tree problem where the edges of a graph are the elements and the indepedent sets are forests in the graph. Kruskal's greedy algorithm is the natural greedy MST algorithm.

More general independence systems

There are many equivalent ways to define matroids. In particular, the exchange property immediately implies that in a matroid M every maximal independent set (*base*) has the same cardinality, the *rank* of M. We can also define a base for any subset $S \subseteq U$. Matroids are those independence systems where all bases have the same cardinality.
A (Jenkyns) k-independence system satisfies the weaker property that for any set S and two bases B and B' of S, $\frac{|B|}{|B'|} \leq k$. Matroids are precisely the case of $k = 1$.
Examples:

- The intersection of k matroids
- Mestre's k-extendible systems where the matroid exchange property is replaced by : If $S \subseteq T$ and $S \cup \{u\}$ and T are independent, then $\exists Y \subseteq T - S : |Y| \leq k$ and $T - Y \cup \{u\}$ is independent.
- Independent sets in $k + 1$ claw free graphs. In such graphs, the neighbourhood of every node has at most k independent vertices.

The standard greedy algorithm for k-systems and $k+1$ claw free graphs

Jenkyns shows that the standard greedy algorithm is a k-approximation for maximizing a linear function subject to independence in a k-independence system. It follows that the standard greedy algorithm is a k-approximation for independence in a $k+1$ claw free graph.
This implies constant approximations for many classes of graphs, in particular for many types of graphs induced by intersections of geometric objects.

Monotone submodular function maximization

- As previously mentioned, the monotone problem is only interesting when the submodular maximization is subject to some constraint.
- Probably the simplest and most widely used constraint is a cardinality constraint; namely, to maximize $f(S)$ subject to $|S| \leq k$ for some k and since f is monotone this is the same as the constraint $f(S) = k$.
- Following Cornuéjols, Fisher and Nemhauser [1977] (who study a specific submodular function), Nemhauser, Wolsey and Fisher [1978] show that the standard greedy algorithm achieves a $1 - \frac{1}{e}$ approximation for the cardinality constrained monotone problem. More precisely, for all k, the standard greedy is a $1 - (1 - \frac{1}{k})^k$ approximation for a cardinality k constraint.

Standard greedy for submodular functions wrt cardinality constraint

$S := \varnothing$
While $|S| < k$
  Let u maximize $f(S \cup \{u\}) - f(S)$
  $S := S \cup \{u\}$
End While

Generalizing to a matroid constraint

- Nemhauser and Wolsey [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. Matroids are an elegant abstraction of independence in a variety of settings.
- Fisher, Nemhauser and Wolsey show that both the standard greedy algorithm and the 1-exchange local search algorithm achieve a $\frac{1}{2}$ approximation for an arbitrary matroid constraint.
- They also showed that this bound was tight for greedy and for the 1-exchange local search.

Matroids and independence systems

- Independence systems and matroids
  Let $M = (U, \mathcal{F})$, where U is a set of elements, $\mathcal{F} \subseteq 2^{|U|}$; $I \in \mathcal{F}$ is called an independent set.
  An (heriditary) independence system satisfies the following properties:
  1) $\varnothing \in \mathcal{F}$; often stated although not necessary if $\mathcal{F} \neq \varnothing$
  2) $S \subseteq T, T \in \mathcal{F} \Rightarrow S \in \mathcal{F}$
- A matroid is an independence system that also satisfies:
  3) $S, T \in \mathcal{F}, |S| < |T|$, then $\exists x \in T \setminus S$ such that $S \cup \{x\} \in \mathcal{F}$
- Sets having at most k elements constitute the independent sets in a uniform matroid
- Other common examples, include
  1. partition matroids where U is the disjoint union $U_1 \cup U_2 \ldots \cup U_r$ and there are individual cardiality constraints k_i for each block U_i of the partition.
  2. Graphic matroids where U is the set of edges E in a graph $G = (V, E)$ and $E' \subseteq E$ is independent if $G = (V, E')$ is acyclic.
  3. Linear matroids where U is a set of vectors in a vector space and I is independent in the usualy sense of linear independence.

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 bases). 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
  $$\sum_{i=1}^{\ell} [f(S) - f(S \setminus \{t_i\}) \leq f(S)$$

Approximation bound using 1-exchange local search for monotone submodular search

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. 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 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 consiting 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 $w : E \to \Re^{\geq 0}$ 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 (subject to a matroid constraint) 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 $\frac{r-1}{2r-\ell-1} \to \frac{1}{2}$ 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 Max-k-Sat), 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 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.
- The potential function is :

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

The Lovász Local Lemma (LLL)

- Suppose we have a set of "bad" random events $E_1, \ldots, E_m$ with $Prob[E_i] \leq p < 1$ for each i. Then if these events are independent we can easily bound the probability that none of the events has occurred; namely, it is $(1-p)^m > 0$.
- Suppose now that these events are not independent but rather just have limited dependence. Namely suppose that each E_i is dependent on at most r other events. Then the Lovász local Lemma (LLL) states that if $e \cdot p \cdot (r+1)$ is at most 1, then there is a non zero probability that none of the bad events E_i occurred.
- As stated this is a non-constructive result in that it does not provide a joint event in which none of the bad events occured.
- There are a number of applications of LLL including (Leighton, Maggs, Rao) routing, the restricted machines version of the Maxmin "Santa Claus" problem and as we shall now see, solving exact k-SAT under suitable conditions on the clauses.

A somewhat canonical application of the LLL

- Let $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a an exact k CNF formula. From our previous discussion of the exact Max-k-Sat problem and the naive randomized algorithm, it is easy to see that if $m < 2^k$, then F must be satisfiable. ($E[\text{clauases satisfied}] = \frac{2^k-1}{2^k} m > m - 1$ when $m < 2^k$.)
- Suppose instead that we have an arbitrary number of clauses but now for each clause C, at most r other clauses share a variable with C.
- If we let E_i denote the event that C_i is not satisfied for a random uniform assignment and hence having probability $1/(2^k)$, then we are interested in having a non zero probability that none of the E_i occurred (i.e. that F is satisfiable).
- The LLL tells us that if $r + 1 \leq \frac{2^k}{e}$, then F is satisfiable.
- As informally but nicely stated in Gebauer et al [2009]: "In an unsatisable CNF formula, clauses have to interleave; the larger the clauses, the more interleaving is required."

A constructive algorithm for the previous proof of satisfiability

- Here we will follow a somewhat weaker version (for $r \leq 2^k/8$) proven by Moser [2009] and then improved by Moser and G. Tardos [2010] to give the tight LLL bound. This proof was succinctly explained in a blog by Lance Fortnow
- This is a constructive proof in that there is a randomized algorithm (which can be de-randomized) that with high probability (given the limited dependence) will terminate and produce a satisfying assignment in $O(mlogm)$ evaluations of the formula.
- Both the algorithm and the analysis are very elegant. In essence, the algorithm can be thought of as a local search search algorithm and it seems that this kind of analysis (an information theoretic argument using Kolmogorov complexity to bound convergence) should be more widely applicable.

The Moser algorithm

We are given an exact k-CNF formula F with m variables such that for every clause C, at most $r \leq 2^k/8$ other clauses share a variable with C.

Algorithm for finding a satisfying truth assignment

Let τ be a random assignment
Procedure SOLVE
  While there is a clause C not satisfied
    Call FIX(C)
  End While

Procedure FIX(C)
  Randomly set all the variables occuring in C
  While there is a neighbouring unsatisfied clause D
    Call FIX(D)
  End While

Sketch of Moser algorithm

- Suppose the algorithm makes at least s recursive calls to FIX. Then $n + s * k$ random bits describes the algorithm computation up to the s^{th} call at which time we have some true assignment τ'.
- That is, the computation (if it halts in s calls is described by the n bits to describe the initial τ and the k bits for each of the s calls to FIX.
- Using Kolmogorov complexity, we state the fact that most random strings cannot be compressed.
- Now we say that r is sufficiently small if $k - \log r - c > 0$ for some constant c, Then the main idea is to describe these $n + s * k$ bits in a compressed way if s is large enough and r is small enough.

Moser proof continued

- Claim: Any C that is satisfied before Fix(C) is called in SOLVE remains satisfied.
- Claim: Working backwards from τ' we can recover the original $n + s * k$ bits using $n + m \log m + s(\log r + c)$ bits; that is n for τ', $m \log m$ for calls to FIX in SOLVE and $\log r + c$ for each recursive call.
- Note: Here it is not stated, but the algorithm does not always terminate