CSC2420: Algorithm Design, Analysis and Theory
Fall 2022
An introductory (i.e. foundational) level graduate course.

Allan Borodin

November 2, 2022

Week 7

Announcements

- I have posted the start of Assignment 2.

Todays agenda

- We ended the October 18 class having stated and analyzed the oblivious local search algorithm for weighted exact max-2-sat. We then stated the non-oblivious local search algorithm for weighted exact max-2-sat. That is where we will continue the discussion.
- **Warning:** I have been ignoring time complexity and even convergence in discussing local search algorithms. There is usually a simple way to guarantee polynomial time at a negligible loss in the approximation ratio. Namely, we can either insist that every local improvement is a sufficiently good improvement or scale the weights to say be integers.

The non-oblivious local search

- We consider the idea that satisfied clauses in S_2 are more valuable than satisfied clauses in S_1 (because they are able to withstand any single variable change).
- The idea then is to weight S_2 clauses more heavily.
- Specifically, in each iteration we attempt to find a $\tau' \in N_1(\tau)$ that improves the potential function
$$\frac{3}{2}W(S_1) + 2W(S_2)$$
instead of the oblivious $W(S_1) + W(S_2)$.
- More generally, for all k, there is a setting of scaling coefficients $c_1, \dots, c_k$, such that the non-oblivious local search using the potential function $c_1 W(S_1) + c_2 W(S_2 + \dots + c_k W(S_k)$ results in approximation ratio $\frac{2^k-1}{2^k}$ for exact Max-k-Sat.

Sketch of $\frac{3}{4}$ totality bound for the non oblivious local search for Exact Max-2-Sat

- Renaming variables, we can assume that τ is the all true assignment.
- Let $P_{i,j}$ be the weight of all clauses in S_i containing x_j.
- Let $N_{i,j}$ be the weight of all clauses in S_i containing $\bar{x}_j$.
- Here is the key observation for a local optimum τ wrt the stated potential:

  $-\frac{1}{2}P_{2,j} - \frac{3}{2}P_{1,j} + \frac{1}{2}N_{1,j} + \frac{3}{2}N_{0,j} \leq 0$
- Summing over variables $P_1 = N_1 = W(S_1)$, $P_2 = 2W(S_2)$ and $N_0 = 2W(S_0)$ and using the above inequality we obtain $3W(S_0) \leq W(S_1) + W(S_2)$

Some experimental results concerning Max-Sat

- Of course, one wonders whether or not a worse case approximation will actually have a benefit in "practice".
- "In practice", local search becomes more of a "heuristic" where one uses various approaches to escape (in a principled way) local optima and then continuing the local search procedure. Perhaps the two most commonly used versions are Tabu Search and Simulated Annealing.
- Later, we will also discuss methods based on online algorithm and "random walks" and other randomized methods (and their derandomizations). .
- We view these algorithmic ideas as starting points.
- But for what it is worth, here are some 2010 experimental results both for artifically constructed instances and well as for one of the many benchmark test sets for Max-Sat.
- Recent experimental results by Poloczek and Willamson show that various ways to use greedy and local search algorithms can compete (wrt. various test sets) with "state of the art" simulated annealing algorithms and walk-sat algorithms while using much less time.

Experiment for unweighted Max-3-Sat

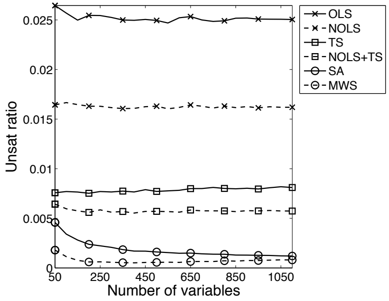

Fig. 1. Average performance when executing on random instances of exact MAX-3-SAT.

[From Pankratov and Borodin]

Experiments for benchmark Max-Sat Instances

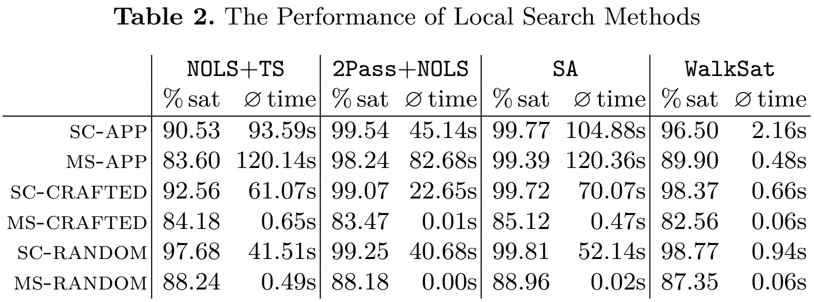

Table 2. The Performance of Local Search Methods

	NOLS+TS		2Pass+NOLS		SA		WalkSat	
	% sat	∅ time	% sat	∅ time	% sat	∅ time	% sat	∅ time
SC-APP	90.53	93.59s	99.54	45.14s	99.77	104.88s	96.50	2.16s
MS-APP	83.60	120.14s	98.24	82.68s	99.39	120.36s	89.90	0.48s
SC-CRAFTED	92.56	61.07s	99.07	22.65s	99.72	70.07s	98.37	0.66s
MS-CRAFTED	84.18	0.65s	83.47	0.01s	85.12	0.47s	82.56	0.06s
SC-RANDOM	97.68	41.51s	99.25	40.68s	99.81	52.14s	98.77	0.94s
MS-RANDOM	88.24	0.49s	88.18	0.00s	88.96	0.02s	87.35	0.06s

Figure: Table from Poloczek and Williamson 2017

Note: *2Pass* is a deterministic "2-pass online algorithm" that is derived from a randomized online algorithm that we will discuss "soon".

Oblivious and non-oblivious local search for $k+1$ claw free graphs

- Consider the k set packing problem and its generalization to $(k+1)$ claw free graphs. (The intersection graph of a k-set instance is a $k+1$ claw free graph where each vertex represents a k-set and there is an edge whenever two sets intersect.)
- We again consider the weighted max (independent) vertex set in a $k+1$ claw free graph. (The greedy and oblivious local search approximations for the weighted k set packing problem generalize to $k+1$ claw free graphs.)
- The standard greedy algorithm and the 1-swap oblivious local search both achieve a k approximation for the WMIS in $k+1$ claw free graphs. Here we define an "ℓ-swap" oblivous local search by using neighbrourhoods defined by bringing in a set S of up to ℓ vertices and removing all vertices adjacent to S.

  NOTE: I am trying to use a convention where we use fractional approximation ratios that are absolute constants and ratios greater than 1 for maximization problems when the ratios are parameterized.

Berman's [2000] non-oblivious local search

- For the unweighted MIS, Halldórsson shows that a a 2-swap oblivious local search will yield a $\frac{1}{2}k + 1$ approximation.
- For the weighted MIS, the "ℓ-swap" oblivous local search results in a $k + \epsilon$ approximation ratio for any constant ℓ where ϵ deopends on ℓ.
- Chandra and Halldóssron [1999] show that by first using a standard greedy algorithm to initialize a solution and then using a "greedy" oblivious local search, the approximation ratio improves to $\frac{2}{3}(k + 1)$
- Can we use non-oblivious local search to improve the locality gap? Once again given two solutions V_1 and V_2 having the same weight, when is one better than the other?

Berman's [2000] non-oblivious local search

- For the unweighted MIS, Halldórsson shows that a a 2-swap oblivious local search will yield a $\frac{1}{2}k + 1$ approximation.
- For the weighted MIS, the "ℓ-swap" oblivous local search results in a $k + \epsilon$ approximation ratio for any constant ℓ where ϵ deopends on ℓ.
- Chandra and Halldóssron [1999] show that by first using a standard greedy algorithm to initialize a solution and then using a "greedy" oblivious local search, the approximation ratio improves to $\frac{2}{3}(k + 1)$
- Can we use non-oblivious local search to improve the locality gap? Once again given two solutions V_1 and V_2 having the same weight, when is one better than the other?
- Intuitively, if one vertex set V_1 is small but vertices in V_1 have large weights that is better than a solution with many small weight vertices.

Berman's [2000] non-oblivious local search

- For the unweighted MIS, Halldórsson shows that a a 2-swap oblivious local search will yield a $\frac{1}{2}k + 1$ approximation.
- For the weighted MIS, the "ℓ-swap" oblivous local search results in a $k + \epsilon$ approximation ratio for any constant ℓ where ϵ deopends on ℓ.
- Chandra and Halldóssron [1999] show that by first using a standard greedy algorithm to initialize a solution and then using a "greedy" oblivious local search, the approximation ratio improves to $\frac{2}{3}(k + 1)$
- Can we use non-oblivious local search to improve the locality gap? Once again given two solutions V_1 and V_2 having the same weight, when is one better than the other?
- Intuitively, if one vertex set V_1 is small but vertices in V_1 have large weights that is better than a solution with many small weight vertices.
- Berman chooses the potential function $g(S) = \sum_{v \in S} w(v)^2$. Ignoring some small ϵ's, his k-swap non-oblivious local search achieves an approximation ratio of $\frac{1}{2}(k + 1)$ for WMIS on $k + 1$ claw-free graphs.

A little more detail on the Chandra and Halldóssron greedy + local search algorithm

For the oblivious local search results when we didn't care about the choice of the initial solution, it didn't matter which local improvement we make.

Chandra and Halldóssron show that for their $\frac{2}{3}$ ratio result the choice local improvement does matter. They use the best local improvement (i.e., swapping in the best vertex and extending to a maximal independent set).

In contrast, a poor choice of a local improvement won't improve upon the approximation that is achieveed without using a greedy initial solutiom.

They also consider an algorithm ANY_α where the improvement must improve by at least an α factor and show that for a suitable choice of α, the algorithm achieves a ratio $\frac{4(k+1)}{5}$.

They show that this $\frac{4}{5}$ ratio is asymptiocally the best ratio for any α.

NOTE: I am not aware of other theoretical results showing how an initial greedy (or random) solution can impove the local search guarantee.

The (metric) facility location and k-median problems

- Two extensively studied problems in operations research and CS algorithm design are the related uncapacitated facility location problem (UFL) and the k-median problem. In what follows we restrict attention to the (usual) metric case of these problems defined as follows:

Definition of UFL

Input: (F, C, d, f) where F is a set of faciltites, C is a set of clients or cities, d is a metric distance function over $F \cup C$, and f is an opening cost function for facilities.
Output: A subset of facilities F' minimizing $\sum_{i \in F'} f_i + \sum_{j \in C} d(j, F')$ where f_i is the opening cost of facility i and $d(j, F') = min_{i \in F'} d(j, i)$.

- In the capacitated version, facilities have capacities and cities can have demands (rather than unit demand). The constraint is that a facility can not have more assigned demand than its capacity so it is not possible to always assign a city to its closest facility.

UFL and k-median problems continued

Deifnition of k-median problem

Input: (F, C, d, k) where F, C, d are as in UFL and k is the number of facilities that can be opened.
Output: A subset of facilities F' with $|F'| = k$ minimizing $\sum_{j \in C} d(j, F')$

- These problems are clearly well motivated. Moreover, they have been the impetus for the development of many new algorithmic ideas.
- There are many variants of these problems and in many papers the problems are defined so that $F = C$; that is, any city can be a facility. If a solution can be found when F and C are disjoint then there is a solution for the case of $F = C$.

UFL and k-median problems continued

- It is known (Guha and Khuller) that UFL is hard to approximate to within a factor better than 1.463 assuming NP is not a subset of $DTIME(n^{\log \log n})$ and the k-median problem is hard to approximate to within a factor better than $1 + 1/e \approx 1.736$ (Jain, Mahdian, Saberi).
- The UFL problem is better understood than k-median. After a long sequence of improved approximation results the current best polynomial time approximation is 1.488 (Li, 2011).
- For k-median, until recently, the best approximation was by a local search algorithm. Using a p-flip (of facilities) neighbourhood, Arya et al (2001) obtain a $3 + 2/p$ approximation which yields a $3 + \epsilon$ approximation running in time $O(n^{2/\epsilon})$.
- Li and Svennsson (2013) have obtained a $(1 + \sqrt{3} + \epsilon)$ approximation running in time $O(n^{1/\epsilon^2})$. Surprisingly, they show that an α approximate "pseudo solution" using $k + c$ facilities can be converted to an $\alpha + \epsilon$ approximate solution running in $n^{O(c/\epsilon)}$ times the complexity of the pseudo solution.

Some additional comments on local search

- An interesting (but probably difficult) open problem is to use a non oblivious local search for either the UFL or k-median problems.
- But suffice it to say now that local search is the basis for many practical algorithms, especially when the idea is extended by allowing some well motivated ways to escape local optima (e.g. simulated annealing, tabu search) and combined with other paradigms.
- Although local search with all its variants is viewed as a great "practical" approach for many problems, local search is not often theoretically analyzed. It is not surprising then that there hasn't been much interest in formalizing the method and establishing limits.
- LP is itself often solved by some variant of the simplex method, which can also be thought of as a local search algorithm, moving fron one vertex of the LP polytope to an adjacent vertex.
    - No such method is known to run in polynomial time in the worst case.

More general independence systems

As we noted, there are many equivalent ways to define matroids. In particular, the exchange property immediately implies that in a matroid M every maximal independent set (called a *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. Let k be a positive integer. 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 $\frac{1}{k}$-approximation for maximizing a linear function subject to independence in a k-independence system. It follows (as we already know from our earlier study of greedy algorithms) that the standard greedy algorithm is a $\frac{1}{k}$-approximation for independence in a $k+1$ claw free graph.

The same approximation applies for a 1-exchange local search algorithm.

Submodular functions

Let U be a universe. In what follows, we will only be interested in set functions that satisfy $f(S) \geq 0$ for all $S \subseteq U$. We will also assume functions are *normalized* in that $F(\varnothing) = 0$, These assumptions are not that essental but they are standard and without these assumptions statements and proofs become somewwhat more complex.

- A *sublinear set function* satisfies the property that $f(S \cup T) \leq f(S) + f(T)$ for all subsetes S, T of U.
- When $f(S \cup T) + f(S \cap T) = f(S) + f(T)$, the function is a linear (also called modular) function.
- A *submodular set function* $f : U \rightarrow \mathbb{R}$ satisfies the following property: $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$
- It follows that modular set functions are submodular and submodular functions are sublinear.
- Submodular functions can be monotone or non-monotone. A monotone submodular function also satisifes the property that $f(S) \leq f(T)$ whenever $S \subseteq T$.

An alternative characterization and examples of submodular functions

Submodular functions satisfy and can also be defined as those satisfying a *decreasing marginal gains* property. Namely,
For $S \subset T, f(T \cup \{x\} - f(T) \leq f(S \cup \{x\}) - f(S)$. That is, adding additional elements has decreasing (more precisely, non increasing) marginal gain for larger sets.

Most applications of submodular functions are for monotone submodular functions. For example, in practice, when we are obtaining results from a search engine, as we obtain more and more results, we tend to obtain less additional value.

Modular functions are monotone.

The rank function of a matroid is a monotone submodular function.

The two most common examples of non-monotone submodular functions are max-cut and max-di-cut (i.e., max directed cut)

Monotone submodular function maximization

- 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**
```

Proof: greedy approx for monotone submodular maximization subject to cardinality constraint

We want to prove the $1 - (1 - \frac{1}{k})^k$ approximation bound.
Let S_i be the set after i iterations of the standard greedy algorithm and let $S^* = \{x_1, \ldots, x_k\}$ be an optimal seti so that $OPT = f(S^*)$. For any set S and element x, let $f_S(x) = f(S \cup \{x\}) - f(S)$ be the marginal gain by adding x to S. The proof uses the following sequence of inequalities:

$f(S^*) \leq f(S_i \cup S^*)$ by monotonicity

$\leq f(S_i) + (f(S_i \cup \{x_1\}) - f(S_i)) + (f(S_i) \cup \{x_1, x_2\} - f(S_i \cup \{x_1\})) + \ldots$

(by submodularity; question 5(a) on assignment)

$\leq f(S_i) + f_{S_i}(x_1) + f_{S_i}(x_2) + \ldots f_{S_i}(x_k)$

(again by submodularity)

$\leq f(S_i) + k \cdot (f(S_{i+1} - f(S_i))$ by the greedy assumption

Equivalently, $f(S_{i+1} \geq f(S_i) + \frac{1}{k}(f(OPT) - f(S_i)$

The proof is completed by showing $f(S_i) \geq (1 - (1 - \frac{1}{k})^i) \cdot OPT$ by induction on i.

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

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 funstion.
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

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

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 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 $w : E \rightarrow \Re^{\geq 0}$ 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 (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 $\frac{r-1}{2r-\ell-1} \rightarrow \frac{1}{2}$ 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 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[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.
- An informal, but nicely stated comment in Gebauer et al [2009] states: "In an unsatisfiable 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 $\tau$ 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

An old but new topic: randomized algorithms

Our next theme will be randomized algorithms. Of course we have already seen randomization in a few online algorithms. However, for the main part, our previous themes have been on algorithmic paradigms, so far online algorithms, variants of greedy and local-search and primal dual algorithms. Randomization is not per se an algorithmic paradigm (in the same sense as greedy algorithms, DP, local search, LP rounding, primal dual algorithms).

An old but new topic: randomized algorithms

Our next theme will be randomized algorithms. Of course we have already seen randomization in a few online algorithms. However, for the main part, our previous themes have been on algorithmic paradigms, so far online algorithms, variants of greedy and local-search and primal dual algorithms. 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 insights 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 randomization in the more standard offline algorithm setting, as we have already seen, randomization plays a very central role in online algorithms as the online setting is particularly vulnerable to worst case adversarial examples. Here are some results 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 Buchbinder et al two sided online greedy algorithm for the unconstrined maximization of a non-monotone submodular function. and application to max-sat.
4. Online with advice and relation to randomized online algorithms
5. De-randomization using two and multi pass algorithms

But first a few more comments on randomization and complexity 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 algorthms 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.

- The naive randomized algorithm can be implemented in a "model 0"

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 Johnson's algorithm in the ROM model is at most $2\sqrt{15}$–$7 \approx .746 < 3/4$, 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.

Yannakakis' IP/LP *randomized rounding* algorithm for Max-Sat

- We will formulate the weighted Max-Sat problem as a $\{0,1\}$ IP.
- Relaxing the variables to be in $[0,1]$, we will treat some of these variables as probabilities and then round these variables to 1 with that probability.
- Let F be a CNF formula with n variables $\{x_i\}$ and m clauses $\{C_j\}$. The Max-Sat formulation is :
  maximize $\sum_j w_j z_j$
  subject to $\sum_{\{x_i \text{ is in } C_j\}} y_i + \sum_{\{\bar{x}_i \text{ is in } C_j\}} (1 - y_i) \geq z_j$
  $y_i \in \{0,1\}; \; z_j \in \{0,1\}$
- The y_i variables correspond to the propositional variables and the z_j correspond to clauses.
- The relaxation to an LP is $y_i \geq 0$; $z_j \in [0,1]$. Note that here we cannot simply say $z_j \geq 0$.

Randomized rounding of the y_i variables

- Let $\{y_i^*\}, \{z_j^*\}$ be the optimal LP solution,
- Set $\tilde{y}_i = 1$ with probability y_i^*.

Theorem

Let C_j be a clause with k literals and let $b_k = 1 - (1 - \frac{1}{k})^k$. Then $Prob[C_j$ is satisifed $]$ is at least $b_k z_j^*$.

- The theorem shows that the contribution of the j^{th} clause C_j to the expected value of the rounded solution is at least $b_k w_j$.
- Note that b_k converges to (and is always greater than) $1 - \frac{1}{e}$ as k increases. It follows that the expected value of the rounded solution is at least $(1 - \frac{1}{e})$ LP-OPT $\approx .632$ LP-OPT.
- Taking the max of this IP/LP and the naive randomized algorithm results in a $\frac{3}{4}$ approximation algorithm that can be derandomized. (The derandomized algorithm will still be solving LPs.)

Unconstrained (non monotone) submodular maximization

- Feige, Mirrokni and Vondrak [2007] began the study of approximation algorithms for the unconstrained non monotone submodular maximization (USM) problem establishing several results:
  1. Choosing S uniformly at random provides a $1/4$ approximation.
  2. An oblivious local search algorithm results in a $1/3$ approximation.
  3. A non-oblivious local search algorithm results in a $2/5$ approximation.
  4. Any algorithm using only value oracle calls, must use an exponential number of calls to achieve an approximation $(1/2 + \epsilon)$ for any $\epsilon > 0$.
- The Feige et al paper was followed up by improved local search algorithms by Gharan and Vondrak [2011] and Feldman et al [2012] yielding (respectively) approximation ratios of .41 and .42.
- The $(1/2 + \epsilon)$ inapproximation (assuming an exponental number of value oracle calls), was augmented by Dobzinski and Vondrak showing the same bound for an explicitly given instance under the assumption that $RP \neq NP$.

The Buchbinder et al (1/3) and (1/2) approximations for USM

In the FOCS [2012] conference, Buchbinder et al gave an elegant linear time deterministic $1/3$ approximation and then extend that to a randomized $1/2$ approximization. The conceptually simple form of the algorithm is (to me) as interesting as the optimality (subject to the proven inapproximation results) of the result. Let $U = u_1, \ldots u_n$ be the elements of U *in any order*.

The deterministic 1/3 approximation for USM

```
$X_0 := \varnothing$; $Y_0 := U$
For $i := 1 \ldots n$
  $a_i := f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$; $b_i := f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$
  If $a_i \geq b_i$
    then $X_i := X_{i-1} \cup \{u_i\}$; $Y_i := Y_{i-1}$
    else $X_i := X_{i-1}$; $Y_i := Y_{i-1} \setminus \{u_i\}$
  End If
End For
```

The randomized 1/2 approximation for USM

- Buchbinder et al show that the "natural randomization" of the previous deterministic algorithm achieves approximation ratio $1/2$.
- That is, the algorithm chooses to either add $\{u_i\}$ to X_{i-1} with probability $\frac{a'_i}{a'_i+b'_i}$ or to delete $\{u_i\}$ from Y_{i-1} with probability $\frac{b'_i}{a'_i+b'_i}$ where $a'_i = \max\{a_i, 0\}$ and $b'_i = \max\{b_i, 0\}$.
- If $a_i = b_i = 0$ then add $\{u_i\}$ to X_{i-1}.
- Note: Part of the proof for both the deterministic and randomized algorithms is the fact that $a_i + b_i \geq 0$.
- This fact leads to the main lemma for the deterministic case:

$$f(OPT_{i-1} - f(OPT_i) \leq [f(X_i - f(X_{i-1}] + [f(Y_i) - f(Y_{i-1}]$$

Here $OPT_i = (OPT \cup \{X_i\}) \cap Y_i$ so that OPT_i coincides with X_i and Y_i for elements $1, \ldots i$ and coincides with OPT on elements $i+1, \ldots, n$. Note that $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$. That is, the loss in OPTs value is bounded by the total value increase in the algorithm's solutions.

Applying the algorithmic idea to Max-Sat

Buchbinder et al are able to adapt their randomized algorithm to the Max-Sat problem (and even to the Submodular Max-Sat problem). So assume we have a monotone normalized submodular function f (or just a linear function as in the usual Max-Sat). The adaption to Submodular Max-Sat is as follows:

- Let $\phi : X \rightarrow \{0\} \cup \{1\} \cup \varnothing$ be a standard partial truth assignment. That is, each variable is assigned exactly one of two truth values or not assigned.
- Let $\mathcal{C}$ be the set of clauses in formula Ψ. Then the goal is to maximize $f(\mathcal{C}(\phi))$ where $\mathcal{C}(\phi)$ is the sat of formulas satisfied by ϕ.
- An extended assignment is a function $\phi' : X \rightarrow 2^{\{0,1\}}$. That is, each variable can be given one, two or no values. (Equivalently $\phi' \subseteq X \times \{0, 1\}$ is a relation.) A clause can then be satisfied if it contains a positive literal (resp. negative literal) and the corresponding variable has value $\{1\}$ or $\{0, 1\}$ (resp. has value $\{0\}$ or $\{0, 1\}$.
- $g(\phi') = f(\mathcal{C}(\phi'))$ is a monotone normalized submodular function. ‘

Buchbinder et al Submodular Max-Sat

Now starting with $X_0 = X \times \varnothing$ and $Y_0 = Y \times \{0, 1\}$, each variable is considered and set to either 0 or to 1 (i.e. a standard assignment of precisely one truth value) depending on the marginals as in USM problem.

Algorithm 3: RandomizedSSAT(f, Ψ)

```
1  $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N} \times \{0,1\}$.
2  for $i$ = $1$ to $n$ do
3      $a_{i,0} \leftarrow g(X_{i-1} \cup \{u_i, 0\}) - g(X_{i-1})$.
4      $a_{i,1} \leftarrow g(X_{i-1} \cup \{u_i, 1\}) - g(X_{i-1})$.
5      $b_{i,0} \leftarrow g(Y_{i-1} \setminus \{u_i, 0\}) - g(Y_{i-1})$.
6      $b_{i,1} \leftarrow g(Y_{i-1} \setminus \{u_i, 1\}) - g(Y_{i-1})$.
7      $s_{i,0} \leftarrow \max\{a_{i,0} + b_{i,1}, 0\}$.
8      $s_{i,1} \leftarrow \max\{a_{i,1} + b_{i,0}, 0\}$.
9      with probability $s_{i,0}/(s_{i,0} + s_{i,1})^*$ do:
       $X_i \leftarrow X_{i-1} \cup \{u_i, 0\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 1\}$.
10     else (with the compliment probability
       $s_{i,1}/(s_{i,0} + s_{i,1})$) do:
11     $X_i \leftarrow X_{i-1} \cup \{u_i, 1\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 0\}$.
12 return $X_n$ (or equivalently $Y_n$).
```

* If $s_{i,0} = s_{i,1} = 0$, we assume $s_{i,0}/(s_{i,0} + s_{i,1}) = 1$.

Further discussion of the Unconstrained Submodular Maximization and Submodular Max-Sat algorithms

- The Buchbinder et al [2012] online randomized $1/2$ approximation algorithm for Unconstrained Submodular Maximization (USM) cannot be derandomized into a "similar" deterministic online or priority style algorithm by a result of Huang and Borodin [2014]. Like the Poloczek result, we claimed that this was "in some sense" evidence that this algorithm cannot be derandomized.
- Their algorithm is shown to have a $\frac{3}{4}$ approximation ratio for Monotone Submodular Max-Sat.
- Poloczek et al [2017] show that the Buchbinder et al algorithm turns out to be equivalent to a previous Max-Sat algorithm by van Zuylen.

The randomized (weighted) max-sat $\frac{3}{4}$ approximation algorithm

The idea of the algorithm is that in setting the variables, we want to balance the weight of clauses satisfied with that of the weight of clauses that are no longer satisfiable.

Let S_i be the assignment to the first i variables and let SAT_i (resp. $UNSAT_i$) be the weight of satisfied clauses (resp., clauses no longer satisfiable) with respect to S_i. Let $B_i = \frac{1}{2}(SAT_i + W - UNSAT_i)$ where W is the total weight of all clauses.

The randomized (weighted) max-sat $\frac{3}{4}$ approximation algorithm

The idea of the algorithm is that in setting the variables, we want to balance the weight of clauses satisfied with that of the weight of clauses that are no longer satisfiable.

Let S_i be the assignment to the first i variables and let SAT_i (resp. $UNSAT_i$) be the weight of satisfied clauses (resp., clauses no longer satisfiable) with respect to S_i. Let $B_i = \frac{1}{2}(SAT_i + W - UNSAT_i)$ where W is the total weight of all clauses.

The algorithm's plan is to randomly set variable x_i so as to increase $\mathbb{E}[B_i - B_{i-1}]$.

The randomized (weighted) max-sat $\frac{3}{4}$ approximation algorithm

The idea of the algorithm is that in setting the variables, we want to balance the weight of clauses satisfied with that of the weight of clauses that are no longer satisfiable.

Let S_i be the assignment to the first i variables and let SAT_i (resp. $UNSAT_i$) be the weight of satisfied clauses (resp., clauses no longer satisfiable) with respect to S_i. Let $B_i = \frac{1}{2}(SAT_i + W - UNSAT_i)$ where W is the total weight of all clauses.

The algorithm's plan is to randomly set variable x_i so as to increase $\mathbb{E}[B_i - B_{i-1}]$.

To that end, let t_i (resp. f_i) be the value of $w(B_i) - w(B_{i-1})$ when x_i is set to true (resp. false).

The randomized max-sat approximation algorithm continued

```
For $i = 1 \dots n$
    If $f_i \leq 0$, then set $x_i =$ true
    Else if $t_i \leq 0$,
      then set $x_i =$ false
    Else set $x_i$ true with probability $\frac{t_i}{t_i + f_i}$.
End For
```

The randomized max-sat approximation algorithm continued

```
For $i = 1 \ldots n$
    If $f_i \leq 0$, then set $x_i = $ true
    Else if $t_i \leq 0$,
      then set $x_i = $ false
    Else set $x_i$ true with probability $\frac{t_i}{t_i + f_i}$.
End For
```

Consider an optimal solution (even an LP optimal) $\mathbf{x}^*$ and let OPT_i be the assignment in which the first i variables are as in S_i and the remiaing $n - i$ variables are set as in $\mathbf{x}^*$. (Note: x^* is not calculated.)

The analysis follows as in Poloczek and Schnitger, Poloczek, and explicitly in Buchbinder et al. One shows the following:

- $t_i + f_i \geq 0$
- $\mathbb{E}[w(OPT_{i-1}) - w(OPT_i)] \leq \mathbb{E}[w(B_i) - w(B_{i-1})]$

The Buchbinder and Feldman derandomization of the USM algorithm

- Contrary to the Poloczek, (resp. Huang and B.) priority inapproximations for Max-Sat (resp. USM), there is another sense in which these algorithms can be derandomized.
- In fact the derandomization becomes an "online algorithm" in the sense that an adversary is choosing the order of the input variables. However rather than creating a single solution, the algorithm is creating a tree of solutions and then takng the best of these.
- The idea is as follows. The analysis of the randomized USM approximation bound shows that a certain linear inequality holds at each iteration of the algorithm. Namely,

  $$E[f(OPT_{i-1} - f(OPT_i)] \leq \frac{1}{2}E[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1}]$$

  That is, the expected change in restricting OPT in an iteration (by setting the i^{th} variable) is bounded by the average change in the two values being maintained by the algorithm.

Continuing the Buchbinder and Feldman derandomization idea

- These inequalities induce two additional inequalties per iteration on the distributions of solutions that can exist at each iteration.
- This then gets used to describe an LP corresponding to these $2i$ constraints we have for the distributions that hold at each iteration of the algorithm.
- But then using LP theory again (i.e. the number of non-zero variables in a basic solution). It follows that we only need distributions with support $2i$ at each iteration rather than the naive 2^i that would follow from just considering the randomized tree.
- Finally, since there must be at least one distribution (amongst the final $2n$ distributions) for which the corresponding solution is at least as good as the expected value. Thus if suffices to take the max over a "small" number of solutions.