Yet Another Example

- Consider a Boolean formula in CNF.
- In CNF, each clause is a disjunction of literals
- The formula is a conjunction of clauses.
- Another name for CNF is Product-of-Sums.

Yet Another Example

- We show that for any set of m clauses, there is a truth assignment that satisfies at least m/2 clauses.
- Proof: Consider a random assignment of truth values to variables as T/F.
- Consider a clause C_i of k variables.
- C_i is not satisfied with probability 2^{-k}.
- Define a random variable Z_i that indicates the event C_i is satisfied.
- $E[Z_i]$ = Pr(Ci is satisfied) = $1 - 2^{-k}$.
- Define Z as the number of clauses satisfied. $Z = \Sigma Z_i$.
- $E[Z] = E[\Sigma Z_i] = \Sigma E[Z_i] = m(1 - 2^{-k}) \geq m/2$ as $k \geq 1$.

Yet Another Example

- We show that for any set of m clauses, there is a truth assignment that satisfies at least m/2 clauses.
- Proof: Consider a random assignment of truth values to variables as T/F.
- $E[Z] = E[\sum Z_i] = \sum E[Z_i] = m(1 - 2^{-k}) \geq m/2$ as $k \geq 1$.
- The above holds irrespective of whether the formula is satisfiable or not.
- The version of the problem where we intend to maximize the number of clauses that can be satisfied is called as MAXSAT.
- MAXSAT is also NP-hard indicating that no good polynomial solutions exist.

Yet Another Example

- The version of the problem where we intend to maximize the number of clauses that can be satisfied is called as MAXSAT.
- Define for an instance I, $m^*(I)$ to be the maximum number of clauses that can be satisfied.
- Let $m^A(I)$ be the number of (expected) clauses that can be satisfied by an (randomized) algorithm A.
- The ratio $m^A(I)/m^*(I)$ is the performance ratio of algorithm A.
- We seek algorithms that this ratio as close to 1.
- The previous approach gives us 1/2 as the ratio.

Yet Another Example

- This version of the problem where we intend to maximize the number of clauses that can be satisfied is called as MAXSAT.
- The ratio $m^A(I)/m^*(I)$ is the performance ratio of algorithm A.
- We seek algorithms that this ratio as close to 1.
- The previous approach gives us 1/2 as the ratio.
  - Actually the ratio is $1-2^{-k}$.
- In fact, there are instances where one can satisfy only 1/2 of the clauses.

LP Rounding

- This version of the problem where we intend to maximize the number of clauses that can be satisfied is called as MAXSAT.
- We now study an approach that does better than 1/2.
- Finally, we devise an algorithm that gets us a ratio of 3/4.

LP Rounding

- The technique of LP Rounding uses the following approach.
- Write the optimization problem as an integer linear program (ILP).
- Relax some of the constraints of the ILP in a step called LP Relaxation to convert the ILP to a simple Linear Program (LP).
- Note that LP can be solved in polynomial time. Get an optimal solution to the LP.
- Round the solution from LP to satisfy the integrality constraints.
- May lose some quality in this step but that is inevitable.

LP Rounding

- Let us apply LP rounding to the MAXSAT problem.
- Consider a clause C_i.
- An indicator variable z_i with values in {0, 1} is defined to indicate whether C_i is satisfied or not.
- We now seek to maximize $\Sigma_i z_i$.
- For each variable x_j, we define an indicator variable y_j that takes values 1 or 0 corresponding to x_j = True or False respectively.
- Since variables can appear in either the pure form or the complemented form, we separate these as follows.

LP Rounding

- Let us apply LP rounding to the MAXSAT problem.
- Consider a clause C_i.
- For each variable x_j, we define an indicator variable yj that takes values 1 or 0 corresponding to x_j = True or False respectively.
- Since variables can appear in either the pure form or the complemented form, we separate these as follows.
- Define C_{i+} to be the indices of variables that appear in pure form in C_i.
- Define C_{i-} to be the indices of variables that appear in pure form in C_i.

LP Rounding

- Let us apply LP rounding to the MAXSAT problem.
- Consider a clause C_i.
- For each variable x_j, we define an indicator variable y_j that takes values 1 or 0 corresponding to x_j = True or False respectively.
- Define C_{i+} to be the indices of variables that appear in pure form in C_i.
- Define C_{i-} to be the indices of variables that appear in pure form in C_i.
- Now, clause C_i is satisfied if it holds that for each i

$$\Sigma_{j \text{ in } C_{i+}} y_j + \Sigma_{j \text{ in } C_{i-}} (1 - y_j) \geq z_i.$$

LP Rounding

- Let us apply LP rounding to the MAXSAT problem.
- The entire integer linear program is:

$$\text{Maximize } \sum_i Z_i$$

subject to

$$\sum_{j \text{ in } C_{i+}} y_j + \sum_{j \text{ in } C_{i-}} (1 - y_j) \geq z_i \text{ for all I}$$

where y_j, z_i in $\{0, 1\}$ for all i and j.

LP Rounding

- Example. Consider the following clauses
- $C_1 = x_1 \vee \neg x_2 \vee x_4$
- $C_2 = x_2 \vee x_3 \vee \neg x_4$
- $C_3 = \neg x_1 \vee x_3$
- $C_4 = \neg x_3 \vee \neg x_4$

and write the corresponding integer linear program and the (relaxed) linear program.

$\neg$: not

$u_1 = 0.2 \quad u_2 = 0.8$

$u_3 = 0.2 \quad v_1 = 0.5$

$0.2 + 0.2 + 0.2 \geq 0.5$

obj: $\max \sum_{i=1}^{4} z_i$

$0.5 + \cdots +$

s.t.

$y_1 + (1 - y_2) + y_4 \geq z_1$

$y_2 + y_3 + (1 - y_4) \geq z_2$

$y_1, y_2, y_3, y_4, z_1, z_2, z_3, z_4 \in \{0, 1\}$

LP

$y_j, z_i \in [0, 1]$

solution: u_j, v_i solution

LP Rounding

- Let us apply LP rounding to the MAXSAT problem.
- Let us relax the constraints on y_j and z_i so that they can take values in [0,1]
- Note they are real numbers between 0 and 1 now and not just integral necessarily.
- We will use u_j and v_i for the values of the best solution to the relaxed linear program.
  - We use u's for the variables and v's for the clauses.
- Notice that $\Sigma_i v_i$ is an upper bound on the number of clauses that can be satisfied.
- But, the values of u_j are not integral, so they do not yet correspond to True/False values in a truth assignment.

LP Rounding

- Let us relax the constraints on y_j and z_i so that they can take values in [0,1]
- We will use u_j and v_i for the values of the best solution to the relaxed linear program.
- But, the values of u_j are not integral, so they do not yet correspond to True/False values in a truth assignment.
- The next step in the technique suggests to round the u_i's so that a truth assignment can be obtained. This step is called randomized rounding.
- Our rounding does the following: Set y_j to 1 with probability u_j.
  - This sets x_j to True with the same probability.

LP Rounding

- The next step in the technique suggests to round the u_j's so that a truth assignment can be obtained. This step is called randomized rounding.
- Our rounding does the following: Set y_j to 1 with probability u_j.
  - This sets x_j to True with the same probability.
- We now estimate the probability that a clause C_i is satisfied.

LP Rounding

- We now estimate the probability that a clause C_i is satisfied.
- Claim: A clause C_i with k literals is satisfied with probability at least $1 - (1-1/k)^k . v_i$.
- Recall what is v_i.
- Let us assume wlog that all the variables in C_i appear in their pure form.
- So, $C_i = x_1 \vee x_2 \vee \ldots x_k$ for some variables x_1 through x_k.
- In the relaxed LP, we satisfy the constraint $u_1 + u_2 + \ldots + u_k \geq v_i$.
- C_i now remains unsatisfied if the corresponding x_1 through x_k are all 0.

LP Rounding

- Claim: A clause C_i with k literals is satisfied with probability at least $1 - (1-1/k)^k . v_i$.
- Recall what is v_i.
- So, $C_i = x_1 \vee x_2 \vee \ldots x_k$ for some variables x_1 through x_k.
- In the relaxed LP, we satisfy the constraint $u_1 + u_2 + \ldots + u_k \geq v_i$.
- C_i now remains unsatisfied if the corresponding x_1 through x_k are all 0.
- This happens with probability $(1-u_j)$ for each variable and hence with probability $\Pi_j (1-u_j)$ for the k variables.
- So, C_i is satisfied with probability $1 - \prod_{j=1}^{k} (1-u_j)$.

↳ $x_1 \quad x_2 \quad \ldots \quad x_k$

F $\quad 1-u_1 \quad 1-u_2 \quad . \quad 1-u_k$

LP Rounding

- Claim: A clause C_i with k literals is satisfied with probability at least $1 – (1-1/k)^k.v_i$.
- This happens with probability $(1-u_j)$ for each variable and hence with probability $\Pi_j (1-u_j)$ for the k variables.
- So, C_i is satisfied with probability $1 – \Pi_j (1-u_j)$.
- We claim that the above is minimized when $u_j = v_i/k$ for each j. (Take the proof as a reading exercise).
- So, the probability of interest is $1 – (1 – v_i/k)^k$.
- We now claim that the function $f(r) = 1 – (1 – r/k)^k$ is at least $1 – (1-1/k)^k.r$ for all r in [0,1].
  - Take the proof of the above also as a reading exercise. You need to show that the function is concave.

LP Rounding

- Claim: A clause C_i with k literals is satisfied with probability at least $1 – (1-1/k)^k.v_i$.
- So, the probability of interest is $1 – (1 – v_i/k)^k$.
- We now claim that the function $f(r) = 1 – (1 – r/k)^k$ is at least $1 – (1-1/k)^k.r$ for all r in [0,1].
- By the above, we conclude that C_i is satisfied with probability at least $(1-1/k)^k.v_i$.
- Now, use linearity of expectations (over clauses) to show that the expected number of satisfied clauses is at least

$$\Sigma_i (1 – (1-1/k)^k).v_i \geq (1 – (1-1/k)^k). \Sigma_i v_i$$
$$\geq (1 – (1-1/k)^k). m^*(I).$$

- Notice that we satisfy at least $(1 – (1-1/k)^k)$-fraction of the maximum number of clauses that can be satisfied.

LP Rounding

- There are two algorithms with expected performance guarantees as shown:

k	$1 - (1/2^k)$	$1 - (1 - 1/k)^k$
1	0.5	1.0
2	0.75	0.75
3	0.875	0.703
4	0.938	0.684
5	0.969	0.672

- As k, the length of the clause, increases, the first algorithm does better, and as k is small, the latter is better.

LP Rounding

- The next steps:
- We have two algorithms. One guarantees an approximation ratio of at least 1/2.
- The other guarantees an approximation ratio of at least
  $1 - (1 - 1/k)^k$.
- We can use both to achieve an approximation ratio of at least 3/4.
- Idea 1 : Run both algorithms and pick the best solution.
- Idea 2 : Toss a fair coin and pick an algorithm based on the outcome.........
- In both cases, we can show that the expected performance ratio will be at least 3/4.

LP Rounding

- We have two algorithms. One guarantees an approximation ratio of at least 1/2. The other guarantees an approximation ratio of at least $1 - (1 – 1/k)^k$.
- We can use both to achieve an approximation ratio of at least 3/4.
- Idea 2 : Toss a fair coin and pick an algorithm based on the outcome.........
- In this case, the expected number of clauses satisfied is $(n_1+n_2)/2$.
  - Here, n_1 and n_2 refer to the expected number of clauses satisfied by the first and the second algorithms.

LP Rounding

- We have two algorithms. One guarantees an approximation ratio of at least 1/2. The other guarantees an approximation ratio of at least $1 - (1 - 1/k)^k$.
- Idea 2 : Toss a fair coin and pick an algorithm based on the outcome.........
- In this case, the expected number of clauses satisfied is $(n_1+n_2)/2$.
  - Here, n_1 and n_2 refer to the expected number of clauses satisfied by the first and the second algorithms.
- We will show that $(n_1+n_2)/2 \geq ¾ \Sigma_i v_i$ $\geq \frac{3}{4} m^*(I)$

LP Rounding

- In this case, the expected number of clauses satisfied is $(n_1+n_2)/2$.
  - Here, n_1 and n_2 refer to the expected number of clauses satisfied by the first and the second algorithms.
- We will show that $(n_1+n_2)/2 \geq \frac{3}{4} \Sigma_j v_j$

$0 \leq v_j \leq 1$

- Focus on clauses that have k literals. Then,
- $n_1 = \Sigma_k \Sigma_{C_j \text{ has k literals}} (1 - 2^{-k}) \geq \Sigma_k \Sigma_{C_j \text{ has k literals}} (1 - 2^{-k}).\ v_j.$
- We know that $n_2 \geq \Sigma_k \Sigma_{C_j \text{ has k literals}}\ \beta_k.\ v_j$ where $\beta_k = 1 - (1 - 1/k)^k$

C_j has k variables

$Pr(C_j \text{ is } T) = (1 - 2^{-k})$

$C_1 \wedge C_2 \wedge C_3 \wedge \cdots \wedge C_m$

C_2 | C_6 C_4 | ... | C_1 C_5

$k=1$ $k=2$ $k=8$

LP Rounding

- We will show that $(n_1+n_2)/2 \geq \frac{3}{4} \Sigma_i v_j$
- Focus on clauses that have k literals. Then,
- $n_1 = \Sigma_k \Sigma_{C_i \text{ has k literals}} (1 - 2^{-k}) \geq \Sigma_k \Sigma_{C_i \text{ has k literals}} (1 - 2^{-k}).\ v_i.$
- We know that $n_2 \geq \Sigma_k \Sigma_{C_i \text{ has k literals}}\ \beta_k.\ v_i$ where $\beta_k = 1 - (1 - 1/k)^k$
- So, $(n_1+n_2)/2 \geq \Sigma_k \Sigma_{C_i \text{ has k literals}} \frac{1}{2} [(1 - 2^{-k}) + \beta_k]\ .\ v_i.$
- Prove that $(1 - 2^{-k}) + \beta_k$ is at least 3/2 for all k. ≥ 1
  - Offline.
- Ending the proof of the required claim.

$\beta_k = 1 - (1 - \frac{1}{k})^k$

$\beta_k \geq \frac{1}{2} + \frac{1}{2^k}$

LP Rounding

- LP Rounding improved the expected performance guarantee for MAXSAT from ½ to ¾.
- There are other techniques developed too that apply to MAXSAT.
  - Semi-definite programming for instance.
- Practical SAT and MAXSAT solvers are also of interest
  - In particular, parallel ones! On a GPU, and scale to a large number of clauses and variables.