CSC373S Lecture 15

- Here is an application of the symbolic determinant problem. Suppose we have an unweighted bipartite graph $G = (V, E)$ with $V = V_1 \cup V_2$, and $E \subseteq V_1 \times V_2$. A matching M is a subset of edges such that no vertex appears in more than one edge in the matching.

  NOTE: We already saw a very restricted case of weighted bipartite matchings when we considered the edit distance problem where the edges in the matching did not intersect. We solved that problem using dynamic programming. What happens when we try to use the same approach for an arbitrary matching?

  Lets assume $|V_1| = |V_2| = n$ and we want to determine if G has a perfect matching (i.e. $|M| = n$). The randomized method we will now provide is not what one would use for a standard sequential computer but it does provide the possibility of an efficient parallel algorithm. (This approach can be extended to provide parallel algorithms for finding a maximum size matching.) Soon we will see how to (sequentially) solve this problem using *max flow*.

  Construct a $n \times n$ symobolic matrix A_G with entries $a_{ij} = x_{ij}$ if $(i, j) \in E$ and 0 otherwise. Consider the symbolic determinant $det(A_G)$. Each non zero term $(-1)^{sgn(\sigma)} \Pi_{i=1}^{n} a_{i,\sigma(i)}$ corresoponds to a perfect matching and since these terms cannot cancel (as polynomials) we conclude that G has a perfect matching iff $det(A_G)$ is not the zero polynomial. As shown in the last lecture we can solve this symbolic determinant problem in $O(n^3)$ time by a randomized algorithm with small one-sided error.

- Our next example of a randomized algorithm pertains to a central problem in logic, AI, and complexity theory. Namely. we will consider the Max-Sat problem. The SAT problem (or CNF-SAT to be more precise) is the first and probably best known NP-complete problem. It is defined as follows: We are given a propositional formula F in conjunctive normal form CNF; that is, F is a conjunction of *clauses* $C_1 \wedge C_2 \ldots \wedge C_m$ where each C_i is a disjunction of literals $\ell_i^{i_1} \vee \ell_i^{i_2} \ldots \vee \ell_i^{i_k}$ and each literal ℓ_i^j is variable x_j or its negation $\overline{x_j}$. The question is to determine whether or not F is satisfiable; i.e. has a truth assignment setting the variables to either TRUE or FALSE such that the resulting formula becomes TRUE.

  F is a k-CNF formula if each clause has at most k literals and we say it is an exact k-CNF formula if each clause has exactly k literals over k distinct variables. The resulting satisfiability question is called the k-SAT (respectively, the exact k-SAT problem) and it is NP-complete for $k \geq 3$. (It is in polynomial time for $k = 2$.)
  We are interested in a corresponding optimization problem Max-SAT and (exact) Max-k-SAT (which are examples of constraint satisfaction problems central to issues in AI). Namely, we want to maximize the number of clauses that can be simultaneously be satisfied. Since this extends the SAT problem, this is obviously an NP-hard optimnization problem for $k \geq 3$ and it turns out to be NP-hard even for $k = 2$.

We shall now see a simple randomized algorithm that will attain (in expectation) a reasonably good approximation of the optimum number of satisfiable clauses. There is also a weighted version of Max-SAT, where each clause C_i has a weight w_i and the goal is to maximize the weight of satisfied clauses. (Here the weight represents the importance of a particular clause = constraint.)

- Lets consider the weighted exact Max-k-Sat problem; that is, we are given $F = C_1 \wedge C_2 \wedge C_m$ where every clause C_j has exactly k literals and has weight w_j. Now what would be the most naive random algorithm?
  Set the truth value of every x_i independetly an uniformly;
  that is, $Prob[\tau(x_i) = TRUE] = Prob[\tau(x_i) = FALSE] = 1/2$.
  Claim: $E_\tau[\sum_{j:C_j is\ satisfed\ by\ \tau} w_j] = \frac{2^k-1}{2^k} \sum_{j=1}^{m} w_j \geq \frac{2^k-1}{2^k} OPT$.
  Here OPT denotes the optimum over all possible truth assignments for the sum of weights of satisfied clauses.

  Proof: A basic fact about expectations is that the expected value of a sum is the sum of the expected values. Therefore, we have $E_\tau[\sum_{j:C_j\ is\ satisfed\ by\ \tau} w_j] = \sum_j E_\tau[C_j\ is\ satisfed\ by\ \tau w_j]$.
  Now we are setting each propositional variable x_i independently so the probability that C_j is not satsified by τ is equal to the probability that each of its k literals were falsified by τ; that is, this happens with probability $\frac{1}{2^k}$. That is, the probability that Cj is satisfied is $\frac{2^k-1}{2^k}$ from which the desired bound on the approximation (to the total weight of all clauses) is obtained.

- Unlike our previous examples for testing polynomial identities, in this case we know how to de-randomize this algorithm and obtain the same approximation. In fact, the deterministic algorithm (called Johnson's algorithm) was known first and then it was recognized that Johnson's algorithm is the de-randomization of the algorithm above. This method of de-randomization is called the method of conditional expectations and it applies to some other randomized algorithms.

  Here is how we can deterministically find a good τ using the method of conditional expectations. In this case, the de-randomized algorithm can be seen to be a greedy (online) alg.

  Consider a computation tree that uniformly and indpendently sets each variable x_i (with $Prob[x_i = TRUE] = Prob[x_i = FALSE] = 1/2$). We can consider the x_i in any order so lets say in the order $x_1, ..., x_n$.

  Let D be the desired expected value.

  Then $D = (1/2)E[F_\tau|\tau(x_1) = TRUE] + (1/2)E[F_\tau|\tau(x_1) = FALSE]$

  Hence, at least one of the two expectations above must be at least D.

  We can compute these two expectations and take the better choice. We do this one variable at a time to determine a τ that has the desired approximation.

- What can be said about Johnson's algorithm when applied to an arbitrary CNF formula (i.e. the Max-SAT problem)? Johnson conjectured (in the mid 70s) that

the approxiamtion bound for his algorithm was 2/3 and after 10 years it was proved that Johnson's algorithm has approximation ratio $\frac{2}{3}$ and that this bound is tight. That is, for all formulas F,
$Johnson(F) \geq \frac{2}{3}OPT$ and there exists a formula F such that $Johnson(F) = \frac{2}{3}OPT$.
Very recently (this January), it has been shown that this approximation ratio improves somewhat if the variables are chosen in a random order. And it has also been shown that a different random truth assignment rule yields an approximation ratio of $\frac{3}{4}$ for all formulas. More sophisticated algorithms (using semi-definite programming) yield improved approximations for Max-2-SAT and Max-SAT.