CS648 : Randomized Algorithms
Semester I, 2011-12, CSE, IIT Kanpur

Random Sampling : part II

Randomized algorithm for Boolean Product Witness Matrix (BPWM)

Random sampling is a very powerful tool to design efficient randomized algorithms. There are various ways and ideas which get materialized by random sampling. In the past we used random sampling for computing approximate median of a set of n numbers in $O(\log n)$ time. There, we exploited the fact that a random sample nearly captures the features of the original set. In particular, we observed that median of a random sample would be an approximate median of the original set.

In today's lecture we discuss another generic use of random sampling. There are many applications where we wish to select a subset of desired properties from a given set. In such applications, it is usually too expensive to compute such set deterministically. However, many times, if we do random sampling in a careful manner, we can get the desired subset with some probability. This application of random sampling may appear a bit vague at this point. But, it will become clear once the reader goes through the complete randomized solution of BPWM problem discussed in this lecture.

1 Problem Statement : BPWM

Given two $n \times n$ Boolean matrices A and B, a matrix C is said to be Boolean product of A and B (to be denoted as $A \times B$) if

$$C_{ij} = \begin{cases} 1 & \text{if } \sum_k A_{ik}B_{kj} > 0 \\ 0 & \text{otherwise} \end{cases}$$

You might be familiar with some fast matrix multiplication algorithms which compute product of a pair of $n \times n$ matrices in time which is subcubic in n. The fastest algorithm for this problem, due to Coppersmith and Winograd takes $O(n^\omega)$ time where $\omega < 2.3716$.

It can be observed that $C_{ij} = 1$ if and only if there exists an integer k such that $A_{ik} = B_{kj} = 1$. Such an index is called a *witness* for the pair (i, j). It can be observed that for each (i, j) with $C_{ij} > 0$ the number of witnesses can be anywhere from 1 to n. Our aim is to compute one witness for each pair (i, j) with $C_{ij} = 1$.

Problem 1.1 **(BPWM)** *Given any two $n \times n$ Boolean matrices A and B, and their Boolean Product Matrix C, compute a witness matrix W defined as follows. W_{ij} stores a witness for (i, j) if $C_{ij} = 1$, and stores 0 otherwise.*

This problem may appear artificial to many, but it has played a key role in solving many problems. In particular, one such problem is all-pairs shortest paths (APSP) problem. Seidel (*J. Comput. Syst. Sci. 51(3): 400-403 (1995)*) showed that for undirected unweighted graphs, we can design a Las-Vegas algorithm to solve APSP in $O(n^\omega \log^2 n)$ time. For this algorithm, an efficient solution of BPWM problem was used in a very crucial manner. The reader may consult the chapter 10 of the book by Motwani and Raghavan for more details.

Note that it can be verified in $O(1)$ time if an integer k is a witness for a pair (i, j). So witness for any pair (i, j), if exists, can be found in $O(n)$ time : make a simultaneous scan of ith row of A and jth column of B and stop as soon as we find k such that $A_{ik} = B_{kj} = 1$. So there is a trivial deterministic algorithm for solving $BPWM$ problem in $O(n^3)$ time. Unfortunately this is also the best know deterministic algorithm for BPWM problem.

So we can see that though computing the product matrix C can be done in subcubic, that is, $O(n^\omega)$ time deterministically, there is no subcubic deterministic algorithm for computing the witness matrix. We shall discuss a Las Vegas randomized algorithm which achieves this goal. The expected running time of this algorithm will be $O(n^\omega \log^2 n)$ time. In short, it employs expected $O(\log^2 n)$ execution of the fast matrix multiplication algorithm to solve BPWM.

2 Some ideas, observations, and the starting point

Let matrix D stores the integer product of the two matrices A and B. That is $D_{ij} = \sum_k A_{ik}B_{kj}$. Can you see that the matrix D stores some information regarding the witnesses ? To know about it, just enquire an entry D_{ij}. What has D_{ij} got to do with witnesses for (i,j)? Well, D_{ij} is in fact the count of all the witnesses for the pair (i,j). So we should try to explore ways to somehow change A or B so that we can compute a witness instead of the count of witnesses for pair (i,j).

Observation 2.1 *We can compute witness for all those pairs (i,j) which have exactly one witness in a single execution of matrix multiplication.*

For the idea underlying the algorithm mentioned in Observation 2.1, have a careful look at Figure 1.

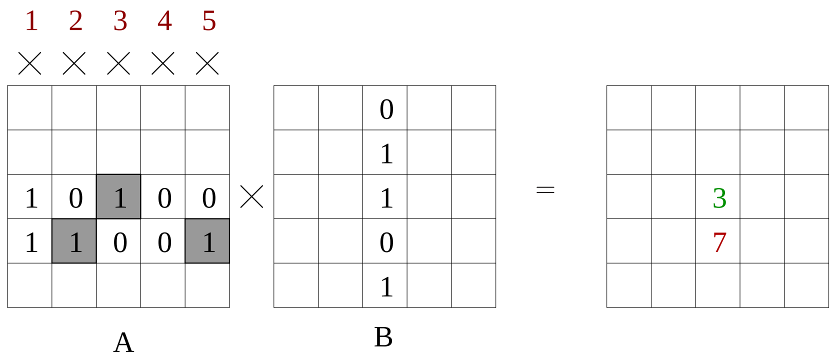

Figure 1: Multiplying kth column by k computes a witness for $(3,3)$ but not for $(4,3)$

So here is the algorithm:

Multiply kth column of A by i. Let A' be the matrix obtained in this manner. Let $D' = A' \times B$. Now consider a pair (i,j) which has exactly one witness, say k. It can be observed that $A'_{ik} = k$. Hence $D'_{ij} = A'_{ik}B_{kj} = k$. Hence for each pair (i,j) which has exactly one witness, D'_{ij} stores the witness.

The above deterministic algorithm is going to be the starting point of our randomized algorithm for BPWM. However, we shall complete our journey to BPWM in a few steps instead of a direct jump. So, let us see whether we can extend the above algorithm for computing a witness for each pair which have exactly t witnesses for a fixed $t > 1$.

2.1 Computing witness for each pair which has exactly t witnesses

Consider a pair (i,j) which has exactly t witness. Let witnesses be $\{k_1, k_2, \cdots, k_t\}$. Now look carefully at the above algorithm and focus on the reason why it does not compute witness for the pair (i,j). Had there been a way to *nullify* the presence of all but one witnesses from $\{k_1, k_2, \cdots, k_t\}$, then the idea of the above algorithm would have worked for pair (i,j) as well. In more precise words, this idea can be expressed as follows.

If we could multiply some columns of A' by zero such that column of exactly one of the witnesses survives, then D'_{ij} would store a witness for (i,j).

There is no deterministic way to materialize the above idea without knowing the set $\{k_1, k_2, \cdots, k_t\}$ itself! Second, and more serious drawback is that there might not be any set which works for all those pairs which have exactly t witnesses. Surprisingly, it will turn out that random sampling overcomes the first hurdle and bypasses the second hurdle. Recall what is conveyed in the first paragraph of this chapter. Does that ring some bell ? Essentially, we want to exploit random sampling to achieve the following objective. The set is $[n]$, and we need a subset of $[n]$ which is likely to have exactly one element from $\{k_1, k_2, \cdots, k_t\}$. It would be desirable to have a reasonably high probability for this event. But firstly let us see the following sampling algorithm which we shall use.

So the question is :

Question 2.1 *What value of p maximizes the likelihood of S having exactly one witness for (i,j) ?*

To answer the above question, we ask a related question : What value of p implies that the expected number of witnesses for pair (i,j) selected in the sample S is 1 (think about the relation between the two)? Applying linearity of expectation, we get the answer $p = 1/t$. The following exercise can be done easily.

```
Algorithm 1: Sample([n], p)
  $S \leftarrow \emptyset$;
  for each $i \in [n]$ do  add $i$ to $S$ with probability $p$;
  return $S$;
```

Exercise 2.1 *For $p = 1/t$, with probability at least $\frac{1}{4}$ exactly one witness for (i, j) is selected in the sample S.*

solution : The exact expression is $\binom{t}{1}\frac{1}{t}(1-\frac{1}{t})^{t-1}$ which is bounded from below by $(1-\frac{1}{t})^t$. Now $(1-\frac{1}{t})^t$ takes value $\frac{1}{4}, \frac{8}{27}, \frac{81}{256}, \cdots$ for $t = 2, 3, 4, \cdots$. In fact it is an increasing function of t and approaches $\frac{1}{e}$ asymptotically. But we are happy with the lower bound $\frac{1}{4}$.

One may feel that $\frac{1}{4}$ probability is too small a probability. Yes, indeed it is small. But there is a way to boost it up to arbitrary close to 1. (Think about how to achieve it. Hint : Exploit the fact that it takes $O(1)$ time to verify if an integer k is a witness for a pair (i, j)). The way to achieve high probability is indeed simple : repeat the sampling multiple times.

Combining all the ideas and observations above, we derive Algorithm 2 which with high probability computes a witness for a pair (i, j) which has t witnesses.

```
Algorithm 2: Witness(t)
  for $\ell = 1$ to $c \log n$ do
      $S \leftarrow$ Sample($[n], \frac{1}{t}$);
      for each $k \in [n]$ do
          if ($k \in S$) then $R_k \leftarrow k$   else  $R_k \leftarrow 0$
      Construct $A'$ such that $A'_{ik} = R_k A_{ik}$;
      $D' \leftarrow A' \times B$;
      for each $(i, j)$ do
          if ($D'_{i,j}$ is witness for $(i, j)$) then $W_{ij} \leftarrow D'_{ij}$
```

Exercise 2.2 *Show by selecting a suitably large but constant value of c that for a pair (i, j) with t witnesses, Algorithm* Witness(t) *fails to find out a witness with probability less than $\frac{1}{n^4}$.*

Though till now, we were focused on just one pair (i, j), we can in fact conclude that the above algorithm in fact computes, with high probability, a witness for all those pairs which have t pairs (Prove it as an exercise instead of looking at the proof that follows. Hint : union theorem).

Lemma 2.1 *Probability that algorithm* Witness(t) *fails to compute a witness for at least one pair with witness count t is bounded by $\frac{1}{n^2}$.*

Proof: Let A be the set of all those pairs $(i, j), 1 \le i, j \le n$ which have exactly t witnesses. Note that $|A| \le n^2$. For a pair $\beta \in A$, let $\mathcal{E}_\beta$ be the event that the algorithm Witness(t) fails to find a witness. Let $\mathcal{E}$ be the event that Witness(t) fails to find a witness for at least one pair from A. Observe that $\mathcal{E} = \cup_{\beta \in A}\mathcal{E}_\beta$. Hence by Union theorem and Exercise 2.2

$$\mathbf{P}[\mathcal{E}] \le \sum_{\beta \in A} \mathbf{P}[\mathcal{E}_\beta] \le \frac{1}{n^4} \le \frac{1}{n^2}$$

But how to employ algorithm Witness(t) to solve BPWM problem. Applying it blindly will give us $O(n^{1+\omega})$ bound, which is inferior to even the trivial deterministic algorithm. What to do now ?

Let us be optimistic and patient at this point and try to think with calm mind.

2.2 Witnesses for all those pairs having witness count in interval $[t, 2t]$

The following question shows us the right way.

Question 2.2 *Won't algorithm* WITNESS(t) *compute, with high probability, witness for all those pairs which have witness count close to t, say in the range* $\in [t, 2t]$ *?*

The answer is yes. Consider any pair (i', j') which has g witnesses and $t \le g \le 2t$. We need to find out the probability that a single execution of the outermost **for** loop of Algorithm WITNESS(t) finds a witness for (i', j'). This probability is the probability that S contains exactly one witness out of g witnesses of (i', j'), and is equal to

$$\binom{g}{1}\frac{1}{t}\left(1-\frac{1}{t}\right)^{g-1} \ge \left(1-\frac{1}{t}\right)^{2t} \ge \frac{1}{16}\ \forall t \ge 2$$

So if we run the outer **for** loop of WITNESS(t) for $4\log_{16/15} n$ times, then with probability at least $1-1/n^4$, Algorithm WITNESS(t) computes a witness for the pair (i', j') which has g witnesses. Extending it to all pairs with witness count in range $[t, 2t]$ is exactly the same as Lemma 2.1. So we can state the following theorem.

Lemma 2.2 *With probability at least* $1 - \frac{1}{n^2}$*, algorithm* WITNESS(2^i) *computes a witness for every pair whose witness count lies in the interval* $[2^i, 2^{i+1}]$ *and runs in time* $O(n^\omega \log n)$*.*

Take a pause here, and try to find out ways to exploit Lemma 2.2 in order to design a Monte Carlo algorithm for BPWM problem with running time $O(n^\omega \log^2 n)$.

2.3 Monte Carlo algorithm for BPWM

Using Lemma 2.2, we just have to execute WITNESS(t) for the each t in the set $\{2^i | i \le \log_2 n\}$. So the overall running time of the algorithm becomes $O(n^\omega \log^2 n)$. However, note that the success probability remains the same, that is, $1 - \frac{1}{n^2}$. Try to give reasons for this fact.

How can you extend the above algorithm to Las Vegas algorithm for BPWM ? You might like to use the fact that it takes $O(1)$ time to verify if an integer k is a witness for a pair (i, j).

3 Points to Ponder

With an open mind and scientific spirit, keep pondering over the following points.

1. How can you extend Monte Carlo algorithm described above to a Las Vegas algorithm for BPWM with expected running time $O(n^\omega \log^2 n)$?

2. Try to go through all the steps we took to design algorithm for BPWM, and try to view the algorithm from all possible angles. Try to realize the role of each tool and importance of each question we used on this way.

3. Try to realize the key role played by random sampling. There are many randomized algorithms which employ random sampling in a similar way as used here in the case of BPWM problem.

4. Try to make a list of useful tips which you learnt from the analysis given above. This might be useful for other problems in future.