Counting Theory Report

Kurt Warren

August 2019

Notation

This past week, on the counting side of things, I worked on solving the theory you mentioned in our meeting. I didn't solve it directly, but sort of had to reformat it a bit to get it to work instead. We will create some specific notation to make writing this a bit easier. Given a formula F, along with a hash function h and cell α, the number of witnesses or solutions of the formula F is defined as $|R_F|$. Moreover, if we consider the cell α among the different cells partitioned by h within the solution set, $|R_{F,h,\alpha|}$ will then denote the number of solutions within that cell. We can then take the i^{th} prefix slice of h, α to obtain a cell of a certain size within the solution set, denoted by $h^{(i)}, \alpha^{(i)}$. Finally, we can denote the number of solutions within a specific cell within the i^{th} prefix slice of h, α by $|R_{F,h^{(i)},\alpha^{(i)}}|$. We will denote this value by $cut(i)$ for shorthand.

Objective and Assumptions

I wanted to show that given a "good" approximation value in the initial phase of the algorithm, we could show that in expectation, each subsequent run in parallel would each only use a constant number of SAT calls. In particular, given the starting hashBits value for the search, the search algorithm will only have to search a constant number of steps away in order to find a good enough cell.

This assumption, as started in the analysis of the algorithm, indicates that $hashBits$ is close to $m = log(|R_F|) - log(pivot)$, where pivot defines the threshold value for each cell. The specific theorem stated that $Pr[m - 2 \cdot log(1 + \epsilon_e) \leq hashBits \leq m] \geq 1 - \delta_e$, where ϵ_e, δ_e are the tolerance and confidence of the initial phase of the algorithm. Since the initial estimation phase is meant to be a rough approximation, we should expect a high ϵ_e value. As such, let's just set it to 1, the max value. Thus, we now have that $Pr[m-2 \leq hashBits \leq m] \geq 1-\delta_e$. Since we can manage the confidence value for this assumption, we can assume this is true and go from there. Moreover, we can work with the value m directly, since $hashBits$ will only ever be a maximum of 2 values away from m.

m is an important value within the log search of the algorithm because it's defined value is $log(|R_F|) - log(pivot)$. Recall that the expectation of the number of solutions in a cell of the i^{th} prefix slice of h, α is $\frac{|R_F|}{2^i}$ (we will denote this value as μ_i). This implies that the expectation of the number of solutions in a cell of the m^{th} prefix slice is $\frac{|R_F|}{2^m} = \frac{|R_F|}{2^{log(|R_F|)-log(pivot)}} = pivot$. Since pivot defines the threshold value for the cells (i.e. $thresh = 1 + pivot \cdot (\frac{\epsilon_r}{1+\epsilon_r}$, a cell with $pivot$ solutions will be highly likely to be the cell returned by the search, since it will have less than $thresh$ solutions, but the cell selected by the previous prefix slice will be greater than thresh, in expectation, as $\mu_i = 2 * \mu_{i+1}$. Moreover, the hash function and bit vector h, α are unchanging, so it must follow that $\forall 1 \leq i \leq n, cut(i+1) \leq cut(i)$.

Initial Proof

Thus, given a cell from the m^{th} prefix slice, we wish to bound the probability that $cut(m)$ or the number of solutions in that specific cell defined by h, α, will be within a certain range of $pivot$. Since one "step" in the value of the prefix slice will change the expectation by a factor of 2, if we were to consider k steps in the prefix-slice, it would be useful to consider the probability as follows:

$$Pr[\frac{1}{2^k} \cdot pivot \leq cut(m) \leq 2^k \cdot pivot] = Pr[\frac{1}{2^k} \cdot pivot \leq cut(m)] \cdot Pr[cut(m) \leq 2^k \cdot pivot]$$

$$= Pr[cut(m) \geq \frac{1}{2^k} \cdot pivot] \cdot Pr[cut(m) \leq 2^k \cdot pivot]$$

Let us consider the first probability. Note that $pivot = \mu_m$, or the expected value of $cut(m)$. Thus, we have that $Pr[cut(m) \geq \frac{1}{2^k} \cdot pivot] = Pr[cut(m) \geq \frac{1}{2^k} \cdot \mu_m]$. Note that given a value $\beta > 0$, the Paley-Zygmund Inequality yields the following bound of $Pr[cut(i) \leq \beta\mu_i] \leq \frac{1}{1+(1-\beta)^2\mu_i}$. We can reverse this probability to obtain that $Pr[cut(i) \geq \beta\mu_i] \geq 1 - \frac{1}{1+(1-\beta)^2\mu_i}$. Substituting i for m, β for $\frac{1}{2^k}$, and μ_m for $pivot$, we obtain that $Pr[cut(m) \geq \frac{1}{2^k} \cdot pivot] \geq 1 - \frac{1}{1+(1-\frac{1}{2^k})^2 pivot}$.

For the second probability, we can do the same with the Markov Inequality. Given a nonnegative random variable X and $a > 0$, recall that the Markov Inequality yields $Pr[X \geq a] \leq \frac{E[X]}{a}$. Let us again reverse this inequality to use for a lower bound. That is, $Pr[X \leq a] \geq 1 - \frac{E[X]}{a}$. We can now apply this inequality to the second desired probability. That is, $Pr[cut(m) \leq 2^k \cdot pivot] \geq 1 - \frac{\mu_m}{2^k \cdot pivot} = 1 - \frac{pivot}{2^k \cdot pivot} = 1 - \frac{1}{2^k}$.

Thus, we now have that $Pr[\frac{1}{2^k} \cdot pivot \leq cut(m) \leq 2^k \cdot pivot] = Pr[\frac{1}{2^k} \cdot pivot \leq cut(m)] \cdot Pr[cut(m) \leq 2^k \cdot pivot] \geq (1 - \frac{1}{1+(1-\frac{1}{2^k})^2 pivot})(1 - \frac{1}{2^k})$. Taking a value such as $k = 3$, for example, yields .87 on the RHS, when the value of $_r$, which

defines *pivot*, is set to .2. However, we still have remember that under the assumption, the value of $hashBits$ is at most 2 less than m.

Next Steps

I'm not quite sure where to go from here. My initial thoughts are to take this bound, and show that we can get from one end of the bound to pivot (for example, from $\frac{1}{2^k} \cdot pivot$ to *pivot*) by taking an additional number of steps with high probability. Then, use the Chebyshev inequality to get the probability that a certain number of solutions will be added (using the standard deviation). For example, maybe k = 3 (so we need to move from $\frac{1}{8}pivot$ to *pivot*), but we take 4 steps. If there is a very high probability that each step will add at least 70% more solutions, then after 4 steps with high probability we should be very close to the pivot value. Then, we would need to just show that with high probability, a constant number of steps away from the pivot value will find the right spot to end the search (which we can figure out using the same method). The only issue with this method is that I'm having a hard time figuring out the standard deviation for taking a step.