Homework 1.3 write-up

Ian Delbridge
CSC 284: Advanced Algorithms
Stefankovic

January 26, 2016

The goal was, given a matrix of random variables' images, to implement an algorithm to find the largest k such that the random variables are k-wise independent. My implementation is in python, and it has very high time complexity though I have made attempts for improvements.
In this discussion, let me refer to the random variables as $X_1, ..., X_n$, the atomic events as $\omega_1, ..., \omega_m$, and the image of X_i as E_i.
I implemented an algorithm that is essentially a brute-force search to find a combination of events such that the following does not hold:

$$Pr(X_{i_1} = a_{i_1} | X_{i_2} = a_{i_2}, ..., X_{i_k} = a_{i_k}) = Pr(X_{i_1} = a_{i_1}) \tag{1}$$

$$\Leftrightarrow \frac{|X_{i_1} = a_{i_1} \wedge ... \wedge X_{i_k} = a_{i_k}|}{|X_{i_2} = a_{i_2} \wedge ... \wedge X_{i_k} = a_{i_k}|} = \frac{|X_{i_1} = a_{i_1}|}{m} \tag{2}$$

$$\Leftrightarrow m|X_{i_1} = a_{i_1} \wedge ... \wedge X_{i_k} = a_{i_k}| = |X_{i_1} = a_{i_1}||X_{i_2} = a_{i_2} \wedge ... \wedge X_{i_k} = a_{i_k}| \tag{3}$$

I formulated the question as a state-search problem to find the closest goal state to the starting position. That is, I described states as tests using (3), with $X_{i_2} = a_{i_2}, ..., X_{i_k} = a_{i_k}$ already known to be independent. Upon finding that (3) holds, because the events are already known to be $k-1$-wise independent, we know that $X_{i_1} = a_{i_1}, X_{i_2} = a_{i_2}, ..., X_{i_k} = a_{i_k}$ are all independent. Thus, the search algorithm I used was simply a breadth-first search. The successors of a state are all states testing the independence of those just found independent and any event created by a random variable not included in the state (i.e. $X_t = a_t$, provided $X_t \notin \{X_{i_1}, ..., X_{i_k}\}$). In this way, the kth level of the search tree contains all of the possible combinations of k events where no two events are generated from the same random variable. My implementation uses a cache to know if a set of events have been intersected before, and retrieving the intersection if so. This intends to eliminate inefficiency of checking a set of events are independent multiple times. In addition, if multiple states are generated that would test that the same set of events is independent, another cache will prevent those states from being expanded.

The running time of the algorithm depends on the number of values that the random variables take on. Using the fact that at the kth level the algorithm has states for each

combination of k events where no two are from the same random variable, we have the following number of states:

$$\frac{(n|E_{i_1}|)((n-1)|E_{i_2}|)...((n-k+1)E_{i_k})}{k!} \tag{4}$$

$$= \frac{n!}{(n-k)!k!} \prod_{j=1}^{k} E_{i_j} \tag{5}$$

The ranges of the random variables have cardinality at most m, the cardinality of the event space, though this is a very high upper bound for independent variables. Therefore, we can bound (5) with

$$\binom{n}{k} \prod_{j=1}^{k} m \tag{6}$$

$$\binom{n}{k} m^k \tag{7}$$

Finally, each state contains an intersection of the events, which is $\mathcal{O}(m)$ time, and the total running time is the sum of all k levels of the tree,

$$\mathcal{O}(m \sum_{i=1}^{k} \binom{n}{i} m^i) \tag{8}$$

To simplify further, use the bound $k \leq n$:

$$\mathcal{O}(m \sum_{i=1}^{n} \binom{n}{n} m^i) \tag{9}$$

$$= \mathcal{O}(m(1+m)^n) = \mathcal{O}((1+m)^n) \tag{10}$$

where we have used the closed form of a binomial sum.

This being said, I find it unlikely to truly use this upper-bounded asymptotic behavior due to the fact that very few sets of n random variables will be n-wise independent. Further, the intersections in consideration become increasingly efficient, because k intersections get progressively smaller.

Improvements for this algorithm may include heuristic searches (if appropriate) to first evaluate sets of events that are more likely to be independent than others. Additionally, one might try a depth-first search to find an l for which the random variables are not l-wise independent for several different sets, and select the smallest l out of these trials, though this will not be deterministic, and I cannot give an estimate for the expected value or the variance of that algorithm.