PROBABILISTIC HEURISTIC SEARCH USING (MULTIVARIATE) ORDER STATISTICS. ILLUSTRATED ON A NETWORK FLOW PROBLEM IN TWO DIMENSIONS

KOBI ABAYOMI,PABLO VANEGAS

Abstract. Abayomi, Fang and Gebraeel [3] (AFG) created a methodology to supplant two heuristic search algorithms on multivariate data with probabilistic versions: the Threshold Algorithm (Fagin [16]) and the Minimal Probing Algorithm [8]).

This advance is motivated by identification of the 'top (bottom) - k' cases in a multivariate system. By exploiting distributional and dependency assumptions the probabilistic algorithm yields a less 'costly' probe the ordered multivariate list. More generally: the probabilistic search algorithms cast a deterministic heuristic search as a probabilistic one via the copula and multivariate order statistics.

Vanegas [19] considered the reallocation of land areas after land development as an Integer Programming (IP) control problem where the objective function is the minimization of sediment flow to riverbeds subject to physical-hydrological constraints via a Network-Flow (NF) formulation. The control variables in Vanegas [19] are calculated from spatial data via raster maps via affine (piecewise-linear) functions; the optimal locations are discovered by Heuristic search.

Here, we replace the deterministic heuristic search - which is used as an alternative to solving the underlying Linear Programming (LP) problem - with a probabilistic one, using the AFG approach. We retain the remainder of Vanegas setup with one exception: we make minimal distribution assumptions on the data that arrive as raster maps and we use these assumptions to generate distributional results on the multivariate list of locations.

Introduction

The setting for this paper is the identification of the best (or worst) k locations in a tableau of n which have $m-1$ attributes - or variables - which contribute to an mth control variable or variable of interest. This is called a 'top-k' search [[4]] and the 'locations' needn't be spatial: any set of similar, not-necessarily identical or independent items - spatial locations, components in a machine...really any system cast as multivariate random variates onto a *scoring function* that generates a list is input to a 'top-k' type algorithm.

The goal is to find the top-k elements in the system using the least amount of information: the fewest machines and minimal number of components in each machine; the fewest locations and minimal number of variates at each location, etc. Generally, this is to sort multivariate data while inspecting as little of it as possible; this is exactly the task in the

Date: June 26, 2015.

All Authors: *Facultad de Ingeneria, Universidad de Cuenca, Cuenca, Ecuador*. Corresponding Author, email: *kobi.abayomi@ucuenca.edu.ec*.

well-known threshold algorithm (TA) [[5]] and its revision, the minimal probing (MPro) [[8]] algorithm - algorithms that are prototypically deterministic in the sorting proceeds without any distributional assumptions on the data and only the most general concatenation of the component information [[15], [16], [13]].

Recent work [[2]] describes an approach to compute a top-k set of answers (analogous to our cases, locations, machines) to an SQL query on a probabilistic database: this set is then sorted by probability. This algorithm runs several Monte-Carlo simulations in parallel, one for each candidate answer, and approximates each probability only to the extent needed to compute correctly the top-k answers. Soliman also [[9]] studies the top-k query in an 'uncertain' database. This paper integrates traditional top-k semantics with "possible worlds" semantics, and constructs a framework that encapsulates a probabilistic model and efficient query processing techniques. While the paper demonstrates a minimal number of accessed m-tuples the algorithms has exponential complexity in both time and component-wise space. In affix to [9], [7] introduces polynomial algorithms for processing top-k queries in uncertain databases under the generally adopted model of x-relations, and the algorithms run in near linear or low polynomial time in uncertain databases.

In [18] and [11] model-driven approaches peculiar to sensor networks are used in conjunction with top-k type query processing. Our technique differs from these in that it is explicitly probabilistic; we rely on multivariate distributional assumptions for the data, given the (particular degradation) model we outline below.

In direct appeal to the TA/MPro algorithms, [10] and [20] have introduced approximate versions via 'guaranteed' thresholding of the error of selecting a non-top-k location. Our approach differs because we explicitly score each location (case, machine), offer general results for the choice of component-wise concatenation/scoring, and yield almost sure convergence to the deterministic result under distributional assumptions.

Cao, Abayomi and Gebraeel [3] generalized these algorithms by exploiting probabilistic information — i.e. explicitly assuming the control variables are data and thus instantiations of random variables — allowing improvement on the ordinary deterministic algorithm by using within item dependency and across item similarity.

We can consider heuristic searches, in this paper as the solution to an LP problem, as the general case and the algorithm employed in Vanegas [2009] as a special case of ranked-list sort of procedures that generate an ordered 'list' as a solution at each iteration, under degenerate probabilistic assumptions.

Our innovation is to recast each of these algorithms under distributional assumptions, and under more general functions for 'scoring' each machine (case, location, whatever), and in this paper cast the solution to the LP problem as a special case of the ranked list algorithm class of methods.

Heuristic Search as the General Case

We call here a *heuristic search* an iterative algorithmic approach to generating a solution to an optimization problem [17].

We introduce this notation: let $\mathbf{a} = (a_1, ..., a_n)$ be a set of *location attributes* which are evaluated on random variates $\mathbf{X}$, where $s_1(\mathbf{X})$ for example is the function which takes the random variates which define the data for the problem and generates the attributes for location 1, of n possible. $\mathbf{X}$ can be of any dimension; as well $s_i(\mathbf{X})$ can be scalar or vector-valued. For simplicity we require

$$a_i(\mathbf{X}) : \mathbb{R}^k \mapsto \mathbb{R}^n \tag{0.1}$$

to be the same k and p $\forall$ i.

Let $\mathbf{a}^*$ be **the** optimal solution set for the location attributes. Let $\mathbf{a}^t$ be values of the location attributes at iteration t of an algorithm. And let $h(\mathbf{a}^t, \mathbf{a}^*) : (\mathbb{R}^n, \mathbb{R}^n) \mapsto \mathbb{R}^n$ be a *heuristic function* that iteratively approximates the next iteration of location attributes such that the distance of the next iterate is not greater than the previous from the optimal solution. That is:

$$\begin{gathered} h(\mathbf{a}^t, \mathbf{a}^*) = \mathbf{a}^{t+1} \\ s.t. \\ d(\mathbf{a}^t, \mathbf{a}^*) \geq d(\mathbf{a}^{t+1}, \mathbf{a}^*) \end{gathered} \tag{0.2}$$

for some distance function d.[1]

The important parts of this abstraction are:

- The separation of the random data from the deterministic attributes via functionalization of the attributes as onto the data.
- The definition of the heuristic function via those attributes.

The attributes $\mathbf{a}$ are cast as deterministic functions on random variates: this allows the specification of these ranked list type algorithms to be fully general; the heuristic function, then, as well is a deterministic onto function subject only to the definition of distance function d, which may specifically be set as $d = h$.

[1]The dimension of this function need not be specified.

Vanegas, Top-'k' & Minimal Probing; Algorithms for Ranked Lists; The special cases

The algorithm in section 3.6 in Vanegas [19] sorts a list of nodes in a directed graph according to values of a control variable. The algorithm enters the best candidates, iteratively and deterministically, to the top of the ordered list, and then proceeds at the next iteration to enter the next best candidates. We will elucidate the algorithm full below after we have constructed the theoretical buttressing for our probabilistic version; here, note merely that the Vanegas algorithm generates an ordered list which is stable and increasing at each iteration.

In this way the Vanegas heuristic differs only from the so-called 'top-k' algorithm in that the calculations 'behind' (if you will) the ordered list are more complex than the scoring function common to the top-k procedure. Both, however, use affine scoring and the stopping rule for both is the same: generate k extremal values of the control variable. Let us introduce the top-k algorithm

Top-'k'. The TA algorithm for top-k ranking was first proposed by Fagin in 1996 [[5]]. The algorithm begins under the assumption that each column of the data is ordered. The algorithm proceeds by scanning the sorted data row-by-row, i.e. case by case (location, machine), and stops when the threshold value of the row is no greater than the minimal overall score of current top-k list. Nevertheless, TA treats the scores as deterministic values, and does not incorporate the distributional information of the scores. Here we develop a probabilistic algorithm by leveraging the dependence among components. This yields a probabilistic TA procedure which converges to the deterministic TA result, but requires less iterations at less than certain levels of tolerance. We recommend either [5] or [6] for a full take on the TA algorithm.

Minimal Probing. The Minimal Probing (MPro) algorithm [[8]] was developed to minimize probe 'cost' for top-k queries. MPro differs from TA chiefly in the relaxation of the sorting prerequisite: in MPro the data are arrive unsorted and the goal is to 'order' as minimally as possible. The MPro algorithm, then, is a recipe for returning the top-k locations (cases, machines) with iterative sorting of particular data. In the deterministic case, MPro is demonstrated to be probe-optimal: i.e. that the prescribed order for inspecting locations has the minimum iterations. Nevertheless, the algorithm is designed for deterministic values and ignores any distributional information for the data. We introduce here a probabilistic version of the MPro algorithm which minimizes the probing 'cost' for the top-k query by explicitly assuming the data are random variables. We illustrate our augmented algorithm (probing strategy) with an associated objective (stopping criterion), based on an explicit assumption about the joint distribution of the data.

The data as random variates

Let real valued $\mathbf{X}$ be the entire data; X_{ij} the (i,j)th variate, without loss of generality let the dimension of $\mathbf{X}$ be $m \times n$.[2] For simplicity we do hold that real vector valued $\mathbf{X}_i$ can follow multivariate normal distribution; this is to assume the vectors are independent from one another (as $i = 1, .., p$) but retain inter-component dependence (as $j = 1, ..., p$). It should be noted that we can consider any elliptical multivariate distribution (multivariate exponential distribution (MVE) or multivariate inverse Gaussian distribution (MIG), for example), i.e. any distribution with second order dependence. The point here is to characterize the data as random, thus probabilistic, and be able to collect - distributionally and parametrically - those properties

Let $X_{(k)j}$ be the k^{th} order statistic for column j such that $X_{(n)j} = \max_{1 \le i \le p} X_{ij}$ and $X_{(1)j} = \min_{1 \le i \le p} X_{ij}$. The goal to derive the distribution for the random vector $\mathbf{X}_{(r)} = [X_{(r)1} X_{(r)2}, ..., X_{(r)m}]$, as well as certain functions of it.

Deriving $F_{X_{(r)}}$, the *c.d.f.* for $X_{(r)}$. For illustration we start with the two-component case. The derivation mimics the problem of putting n balls into 4 bins with constraints, as illustrated in Table 1.

X_{i1}	X_{i2}		Total
	$\le x_2$	$> x_2$	
$\le x_1$	t_1	t_2	l_1
$> x_1$	t_3	t_4	$n - l_1$
Total	l_2	$n - l_2$	n

Table 1 Deriving the distribution of multivariate order statistics when $m = 2$

The derivation:

$$F_{X_{(r)}}(x_1, x_2) = \mathbb{P}(X_{(r)1} \le x_1, X_{(r)2} \le x_2)$$

$$= \mathbb{P}(\text{At least } r \text{ of } X_{i1} \le x_1, \text{at least } r \text{ of } X_{i2} \le x_2)$$

$$= \sum_{l_1, l_2 \ge r} \mathbb{P}(\text{Exactly } l_1 \text{ of } X_{i1} \le x_1, \text{exactly } l_2 \text{ of } X_{i2} \le x_2)$$

[2]The data needn't be square or even matrix.

$$= \sum_{\substack{0 \le t_1, t_2, t_3, t_4 \le n \\ t_1+t_2+t_3+t_4=n \\ r \le l_1, l_2 \le n}} \frac{n!}{t_1!t_2!t_3!t_4!} p_1^{t_1} p_2^{t_2} p_3^{t_3} p_4^{t_4} \tag{0.3}$$

where

$$p_1 = \mathbb{P}(X_{i1} \le x_1, X_{i2} \le x_2), \tag{0.4}$$
$$p_2 = \mathbb{P}(X_{i1} \le x_1, X_{i2} > x_2), \tag{0.5}$$
$$p_3 = \mathbb{P}(X_{i1} > x_1, X_{i2} \le x_2), \tag{0.6}$$
$$p_4 = \mathbb{P}(X_{i1} > x_1, X_{i2} > x_2). \tag{0.7}$$

So, for the special cases of the above result, the *c.d.f.* of the "top row" is

$$F_{X_{(n)}}(x_1, x_2) = p_1^n \tag{0.8}$$

and the *c.d.f.* of the "bottom row" is

$$F_{X_{(1)}}(x_1, x_2) = 1 - (p_2 + p_4)^n - (p_3 + p_4)^n + p_4^n. \tag{0.9}$$

This can immediately be generalized to m-component case following the above outline. Let $c = 2^m$, and the result is as follows

$$\begin{aligned} & F_{X_{(r)}}(x_1, x_2, ..., x_m) \\ =& \mathbb{P}(X_{(r)1} \le x_1, X_{(r)2} \le x_2, ..., X_{(r)m} \le x_m) \\ =& \sum_{\substack{0 \le t_1, t_2, ..., t_c \le n \\ t_1+t_2+..+t_c=n \\ r \le l_1, l_2, ..., l_m \le n}} \frac{n!}{t_1!t_2!...t_c!} p_1^{t_1} p_2^{t_2} ... p_c^{t_c}, \end{aligned} \tag{0.10}$$

(0.11)

where $p_1, p_2, ..., p_c, l_1, ..., l_m$ are defined in the same manner as two-component case.

Deriving the distribution of functions of $X_{(r)}$. After obtaining the *c.d.f.* of $\mathbf{X}_{(r)}$, we need to derive the distribution of scoring functions $\tau(.)$ of $\mathbf{X}_{(r)}$. Typically, the 'scoring' function $\tau(.)$ is a monotone function like max, min, sum [see [6]]. Let G_r denote the *c.d.f.* of $\tau(\mathbf{X}_{(r)})$. When τ is the max function,

$$G_r(s) = \mathbb{P}(\tau(\mathbf{X}_{(r)}) \le s) \tag{0.12}$$
$$= \mathbb{P}(\max_{1\le j\le m} X_{(r)j} \le s)$$
$$= \mathbb{P}(\cap_{j=1}^{m} X_{(r)j} \le s)$$
$$= F_{X_{(r)}}(s, s, ..., s). \tag{0.13}$$

Similarly, when τ is the *min* function,

$$G_r(s) = \mathbb{P}(\tau(\mathbf{X}_{(r)}) \le s) \tag{0.14}$$
$$= \mathbb{P}(\min_{1\le j\le m} X_{(r)j} \le s)$$
$$= \mathbb{P}(\cup_{j=1}^{m} X_{(r)j} \le s)$$
$$= 1 - \mathbb{P}(\cap_{j=1}^{m} X_{(r)j} > s)$$
$$= 1 - \bar{F}_{X_{(r)}}(s, s, ..., s), \tag{0.15}$$

where $\bar{F}_{\mathbf{X}_{(r)}}$ is the survival function of $\mathbf{X}_{(r)}$. For the distribution of the sum, for example, of the r^{th} row, we just adopt the linear transformation on the $X_{(r)}$, then take limit with respect to other variables. This is just the ordinary change of variable method which enables us to get the desired *c.d.f.* for monotone functions $\tau(.)$ of each *rth* row.

From the distribution of $\tau(\mathbf{X}_{(r)})$, we can compute the probability that we have found the "true" top-k locations at each row. Suppose t_r is the minimum overall score of the current top-k list before probing the r^th row, then

$$p_r \triangleq \mathbb{P}(\text{The algorithm stops at } r^{th} \text{ row})$$
$$= \mathbb{P}(\tau(\mathbf{X}_{(r)}) \le t_r) = G_r(t_r). \tag{0.16}$$

Theorem 0.1. Assume $\tau : \mathbb{R}^m \to \mathbb{R}$ is a non-decreasing function. Then p_r is nondecreasing with respect to r.

Proof. When $r \in 1, 2, ..., n-1$ then $r+1 \in 2, ..., n$. As we mine the data 'deeper', we have $t_{r+1} \ge t_r$ since the top-k list will be updated only when the overall score of a 'new' location is larger than that of one of the existing locations. As they are order(ed) statistics, $X_{(r)j}$ is larger than $X_{(r+1)j}$ in stochastic order, $\forall j = 1, ..., m$. Since τ is non-decreasing function, then $\tau(X_{(r)})$ is larger than $\tau(X_{(r+1)})$ in stochastic order [see [12]].

From the definition of stochastic order,

$$
\begin{aligned}
p_r &= \mathbb{P}(\tau(\mathbf{X}_{(r)}) \le t_r) \\
&\le \mathbb{P}(\tau(\mathbf{X}_{(r)}) \le t_{r+1}) \\
&\le \mathbb{P}(\tau(\mathbf{X}_{(r+1)}) \le t_{r+1}) = p_{r+1}.
\end{aligned}
\tag{0.17}
$$

Hence the result holds. □

Using the monotone property of p_r, with a pre-determined threshold confidence level α, we can stop the the algorithm when $p_r \ge \alpha$. In this way, we can find the top-k locations with desired confidence by scanning fewer rows than the TA.

Probabilistic TA algorithm. The algorithm then is a probabilistic augmentation of the top-k ranking algorithm based on TA.

Algorithm

```
(1) Rank each attribute of the
    original
      database in descending
    order
(2) Probe the "sorted" database
    row-by-row from the top,
      and find the current top-k
    locations after probing each
    row
(3) Compute the minimum of
    the overall scores of the
    current
      top-k locations
(4) Repeat step 2 and 3 until
    the probability of finding
    the top-k exceeds the
    threshold, i.e.,
      $\mathbb{P}(g(X_{(r)}) \le t_r) \ge \alpha$.
```

Table 2 Probabilistic Threshold Algorithm

Probabilistic top-k query based on MPro algorithm

The Minimal Probing (MPro) algorithm [[8]] was developed to minimize probe 'cost' for top-k queries. MPro differs from TA chiefly in the relaxation of the sorting prerequisite: in MPro the data are arrive unsorted and the goal is to 'order' as minimally as possible. The MPro algorithm, then, is a recipe for returning the top-k locations (cases, machines) with iterative sorting of particular data. In the deterministic case, MPro is demonstrated to be probe-optimal: i.e. that the prescribed order for inspecting locations has the minimum iterations. Nevertheless, the algorithm is designed for deterministic values and ignores any distributional information for the data. We introduce here a probabilistic version of the MPro algorithm which minimizes the probing 'cost' for the top-k query by explicitly assuming the data are random variables. We illustrate our augmented algorithm (probing strategy) with an associated objective (stopping criterion), based on an explicit assumption about the joint distribution of the data.

Explicitly modeling the data. As before, suppose X_{ij} is the score of the j^{th} component of the i^{th} machine, $X_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$, $i = 1, ..., n, j = 1, ..., m$. Here we need to also make explicit the correlation structure between the components, that is, $Corr(X_{ij}, X_{ij*}) = Corr(X_{kj}, X_{kj*})$ for all i, j, j^*, k.

Thus the distribution of scores of the i^{th} location is $\mathbf{X}_i \sim N(\mu_i, \Sigma_i)$, where

$$\begin{aligned}
&\mu_i = (\mu_{i1}, \mu_{i2}, ..., \mu_{im}),\\
&\Sigma_i = V_i^{1/2} R V_i^{1/2},\\
&V_i = diag(\sigma_{i1}^2, \sigma_{i2}^2, ..., \sigma_{im}^2),\\
&R_{jk} = Corr(X_{ij}, X_{ik}) \triangleq \rho_{jk}.
\end{aligned}$$

Let $X_{ij}^* = \frac{X_{ij}-\mu_{ij}}{\sigma_{ij}}$, then $X_{ij}^* \sim N(0,1)$. If we denote $\mathbf{X}_i^* = (X_{i1}^*, X_{i2}^*, ..., X_{im}^*)$, $\mathbf{X}_i^*$'s will be i.i.d multivariate normal, and the correlation matrix of $\mathbf{X}_i^* = (X_{i1}^*, X_{i2}^*, ..., X_{im}^*)$ is

$$R = \begin{bmatrix} 1 & \cdots & \rho_{1m} \\ \vdots & \ddots & \vdots \\ \rho_{m1} & \cdots & 1 \end{bmatrix}.$$

We can use sample correlation matrix of the standardized variables as our estimate for R. In practice, R is estimated from historical (training) data and could be used for validation.

The essential difference between the MPro and TA algorithms is that the entire row of ordered data is not available in MPro as we iterate. Thus, we need to calculate some sort of conditional information at each iteration and let the locations (cases, machines) enter the 'top-k' set based on this conditional criterion.

***Probing strategy.** For TA, we have to probe the data row-by-row; all components in a row (machine, location, case) are inspected simultaneously. In MPro we can probe the components more flexibly.

Suppose the scoring function is the *sum*, i.e., the overall score of the i^{th} location is

$$\tau_i = \sum_{j=1}^{m} X_{ij}. \tag{0.18}$$

Assume the order of the components to probe is the same for all inspected locations (machines, cases) at each iteation. After probing r components, the conditional expectation is

$$\mathbb{E}(\tau_i|\mathbf{X}_{ir}) = \sum_{j=1}^{r} X_{ij} + \sum_{j=r+1}^{m} \mathbb{E}(X_{ij}|\mathbf{X}_{ir}). \tag{0.19}$$

Since X_{ij} and $\mathbf{X}_{ir}$ are jointly normally distributed,

$$\mathbb{E}(X_{ij}|\mathbf{X}_{ir}) = \mu_{ij} + \Sigma_{ijr}\Sigma_{irr}^{-1}(\mathbf{X}_{ir} - \mu_{ir}),$$

where

$$\begin{aligned}
\mathbf{X}_{ir} &= (X_{i1}, X_{i2}, ..., X_{ir}),\\
\mu_{ir} &= (\mu_{i1}, \mu_{i2}, ..., X\mu_{ir}),\\
\Sigma_{ijr} &= Cov(X_{ij}, \mathbf{X}_{ir}),\\
\Sigma_{irr} &= Var(\mathbf{X}_{ir}).
\end{aligned}$$

The conditional variance is

$$\begin{aligned}
\mathrm{Var}(g_i|\mathbf{X}_{ir}) &= \mathrm{Var}(\sum_{j=r+1}^{m} X_{ij}|\mathbf{X}_{ir})\\
&= Var(Y_i|\mathbf{X}_{ir})\\
&= \Sigma_{iyy} - \Sigma_{iyr}\Sigma_{irr}^{-1}\Sigma_{iyr}',
\end{aligned}$$

where

$$\begin{aligned}
Y_i &= \sum_{j=r+1}^{m} X_{ij},\\
\Sigma_{iyy} &= \mathrm{Var}(Y_i) = \sum_{1\le j,k\le m} Cov(X_{ij}, X_{ik})
\end{aligned}$$

and

$\Sigma_{iyr} = Cov(Y_i, \mathbf{X}_{ir})$

$= (\sum_{j=r+1}^{m} Cov(X_{ij}, X_{i1}),$

$..., \sum_{j=r+1}^{m} Cov(X_{ij}, X_{ir})).$

We define the *probing score* as

$$Sp_i \triangleq \mathbb{E}(\tau_i|\mathbf{X}_{ir}) + \lambda Sd(\tau_i|\mathbf{X}_{ir}), \tag{0.20}$$

where λ is the tuning parameter, and $Sd(\tau_i|\mathbf{X}_{ir}) = \sqrt{\mathrm{Var}(\tau_i|\mathbf{X}_{ir})}$. We use the probing scores decide the order of data inspection: i.e., at each iteration, the algorithm probes the location (machine, case) with the highest probing score unless the location has been completely probed. *The probing score can also be regarded as the upper bound of the confidence interval of the overall score, given the information gained from components that have been probed.* The algorithmic parameter λ we introduce in the probing score Sp should be chosen using training data and can serve to optimize the number of iterations..

A probabilistic stopping criterion for MPro. Recall that $\mathbf{X}_{ir}$ denotes the components of location i that have been probed. We sort the locations ascending according to $\mathbb{E}(\tau_i|\mathbf{X}_{ir})$, and let $W_i = \tau_{[i]}|\mathbf{X}_{[ir]}$ denote the conditional overall score with i^{th} smallest mean. Note that W_i is not an order statistic. Let $Y = \min_{i=n-k+1}^{n} W_i$, and $Z = \max_{i=1}^{n-k} W_i$.

The stopping criterion of the algorithm is

$$\mathbb{P}(Y \geq Z) \geq \alpha \tag{0.21}$$

where α is the pre-determined threshold.

To calculate $\mathbb{P}(Y \geq Z)$, we use $\mathbb{P}(Y \geq Z) = \int_{-\infty}^{\infty} \int_{z}^{\infty} f_y(y) f_z(z) dz dy$, where

$$f_Y(y) = \left(\sum_{i=n-k+1}^{n} \frac{f_i(y)}{\bar{F}_i(y)} \right) \prod_{i=n-k+1}^{n} \bar{F}_i(y), \tag{0.22}$$

$$f_Z(z) = \left(\sum_{i=1}^{n-k} \frac{f_i(z)}{F_i(y)} \right) \prod_{i=1}^{n-k} F_i(y). \tag{0.23}$$

Here $F_i, \bar{F}_i$, and f_i are the c.d.f, survival function and p.d.f. of W_i, respectively. In specific, when W_i is a fixed number, that is, if location i is fully probed, f_i is the Radon-Nikodym derivative of F_i [see [21]].

Algorithm

```
(1) Calculate the probing scores
    $Sp^{(\mathcal{Q})}$ and put them in the
    probing queue $Q.top$, where
    $\mathcal{Q}$ denote the observed
    information.
(2) Probe $Q.top$, update $\mathcal{Q}$.
(3) While the stopping criterion
    is not satisfied, repeat
    step 1 and 2.
```

Table 3 Probabilistic MPro Algorithm

An MPro probabilistic algorithm.

Properties of the algorithm. The augmentation of the MPro algorithm with the stopping criteria for explicitly probabilistic data allows us to demonstrate that the ranking algorithm can stop as the number of iterations increases.

Theorem 0.2. *Define $P_{stop} = \mathbb{P}(Y \geq Z)$, and let n_{pr} denote the total number of probes. Then $P_{stop} \to 1$ as $n_{pr} \to n \cdot m$.*

Proof. When $n_{pr} = n \cdot m$, all the components of all the locations are probed, and all the conditional overall scores are fixed values. Recall that W_i's are conditional overall scores sorted ascending according to their means, then we have $\mathbb{P}(Y \geq Z) = \mathbb{P}(\min_{i=n-k+1}^{n} W_i \geq \max_{i=1}^{n-k} W_i) = 1$. Thus when $n_{pr} \to n \cdot m$, the algorithm stops almost surely. □

As well, the choice of score — using the first and second conditional moments of τ at each iteration — is sufficient for the stopping criteria.

Theorem 0.3. *The probing scores Sp are 'sufficient' to calculate P_{stop}.*

Proof. Y, Z are order statistics of the conditional overall scores. Since the scores are normally distributed, the conditional overall scores also follow the normal distribution. Recall that the probing score Sp is linear combination of conditional expectation and conditional standard deviation. The Sp's are 'sufficient' to calculate P_{stop} in the sense that mean and s.d. are sufficient to characterize normal distribution. □

This is the crux of the probabilistic extension of the TA algorithm: the derivation of the distribution of the multivariate order statistics, the *rth* of n independent, sorted, rows. We use this result to predict the stopping 'place' (iteration) for the Threshold Algorithm, and use that as the objective.

this is referred to as a ‘top-k’ search [[4]]. The goal is to find the top-k locations using the least amount of information: the fewest locations and minimal number of components in each location. More generally, this is to sort multivariate data while inspecting as little of it as possible; this is exactly the task in the well-known threshold algorithm (TA) [[5]] and its revision, the minimal probing (MPro) [[8]] algorithm. These algorithms are prototypically deterministic: the sorting proceeds without any distributional assumptions on the data and only the most general concatenation of the component information [[15], [16], [13]]. Our innovation is to recast these algorithms under distributional assumptions, and under more general functions for ‘scoring’ each machine (case, location).

Locating Best Nodes: Vanegas, P-top-k, P-MPro

We set up the example using the approach in [1] but let:

$$F_{\underset{\sim}{\mathbf{X}}} = C_{\Phi}(F_{XD}, ..., F_{ST}) \tag{0.24}$$

be the joint distribution that we impose via the Gaussian Copula C_{Φ} [14]. This is a necessary restriction on the problem so that we can exploit the 2-dependence of the Gaussian distribution and parameterize our probabilistic queries as above; the copula approach allows us to retain the same marginal distributions as in [1].

The approach we take here is strictly frequentist. This diverges from [1]: the distributions we list here and in (0.24) are not priors - here the priors are degenerate, as such are the posteriors - but **the** distributions for each margin, which we join via the Gaussian Copula.

Let the data be as before:

- Flow Direction: We let each element in

$$\underset{\sim}{X}_{FD} \sim DiscUnif(0, 8) \tag{0.25}$$

  that is a discrete uniform random variable with equal mass on each of the possible flow directions.

- Flow Production: Each

$$\underset{\sim}{X}_{FP,0}, \underset{\sim}{X}_{FP,1} \sim N(0, 1) \tag{0.26}$$

  without loss of generality.

- Flow Factor: Each

$$\underset{\sim}{X}_{FF,0}, \underset{\sim}{X}_{FF,1} \sim Unif(0, 1) \tag{0.27}$$

  without loss of generality.

- Breakpoint 1/Retention Capacity: Each

$$\underset{\sim}{X}_{BP1,0}, \underset{\sim}{X}_{BP1,1} \sim Unif(0,1) \tag{0.28}$$

  without loss of generality.
- Breakpoint 2: Each

$$\underset{\sim}{X}_{BP2,0}, \underset{\sim}{X}_{BP2,1} \sim Unif(0,1) \tag{0.29}$$

  without loss of generality
- Streams: Each

$$\underset{\sim}{X}_{ST} \sim Ber(0,1) \tag{0.30}$$

The same 'toy' data in [1] and [19] are reproduced here. The data are:

$$\underset{\sim}{X}_{FD} = \begin{bmatrix} 5 & 5 & 5 & 0 \\ 4 & 4 & 4 & 3 \\ 4 & 5 & 3 & 3 \\ 4 & 3 & 3 & 2 \end{bmatrix} \tag{0.31}$$

$$\underset{\sim}{X}_{FP,0} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{0.32}$$

$$\underset{\sim}{X}_{FF,0} = \begin{bmatrix} 1 & .5 & .5 & 1 \\ 1 & .4 & .7 & .2 \\ 1 & .2 & .7 & .2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{0.33}$$

$$\underset{\sim}{X}_{BP1,0} = \begin{bmatrix} .5 & .5 & .5 & .5 \\ .5 & .5 & .5 & .5 \\ .5 & .5 & .5 & .5 \\ .5 & .5 & .5 & .5 \end{bmatrix} \tag{0.34}$$

and

$$\underset{\sim}{X}_{BP2,0} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{0.35}$$

Figure 1 Matrix representation of data

These are five 4×4 matrices generating $n = 16$ 'locations'. Another simplification we exploit is to cast the attributes of interest - Effective Accumulation (EA) - as the output of the attribute function we define above in eq. (0.1). The data are also defined by the direction of flow (inferable from the the flow direction matrix) but representable as a directed graph, see Figure 2

Since we are taking a strictly frequentist approach we ignore the distribution of the transformation - section II in [1] and instead cast all of the random variates as location attributes (see eq. (0.1), or in the language of [3], as the score of the multivariate data vector. Here we sublimate the distribution of the remainder of the affine transformations in [1] and focus on Effective Accumulation, which, as an affine transformation of Gaussian random variate - via (0.24) remains Gaussian.

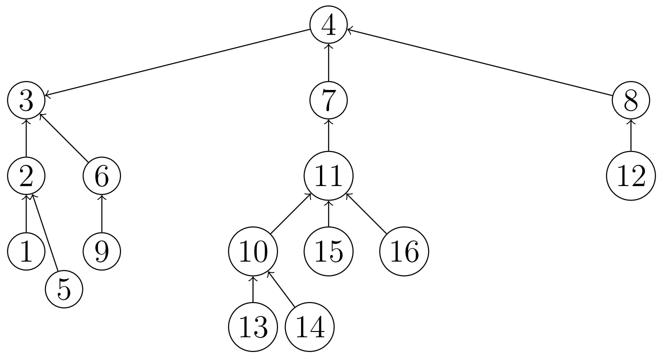

Figure 2 Graphical representation of data in Figure 1

In this example we set $k = 7$ and calculate the probability that a node is in the top-7 candidates for deforestation at each iteration. Note the difference between the TA and MPro algorithms: once a node enters the top-7 in TA it is almost certain to remain in the top-5 as the probabilistic calculation is done list-wise. The MPro approach, on the other hand, has a high probability for each node to remain in the top-5 once entered - but this probability is not certain.

Iteration	1	2	3	4	5	6	7
node 1	▨	▨	▨	▨	▨	🟥	🟥
node 2	🟥	🟥	🟥	🟥	🟥	🟥	🟥
node 3	▨	▨	▨	▨	▨	▨	▨
node 4	▨	▨	▨	▨	🟥	🟥	🟥
node 5	▨	▨	▨	▨	▨	▨	▨
node 6	🟥	🟥	🟥	🟥	🟥	🟥	🟥
node 7	▨	▨	▨	▨	▨	▨	▨
node 8	▨	▨	▨	▨	▨	▨	▨
node 9	▨	▨	▨	▨	▨	▨	▨
node 10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
node 11	▨	▨	▨	▨	▨	▨	▨
node 12	▨	▨	▨	▨	▨	▨	▨
node 13	▨	🟧	🟧	🟧	🟧	🟧	🟥
node 14	▨	🟧	🟧	🟧	🟧	🟧	🟧
node 15	▨	▨	▨	▨	▨	▨	▨
node 16	▨	▨	▨	▨	▨	▨	▨

Figure 3 7 iterations of the Probabilistic TA on the Network Flow problem. Red cells have greater than 75 percent probability of being in top 7; orange greater than 50 percent. The true list is discovered after only 6 iterations.

On this low dimension problem the Probabilistic TA approach outperforms the CAMF heuristic in [19] and the Markov Chain approach in [1]. Once the threshold probability for inclusion in the list of best locations (top 7 in this case) - the list remains stable.

Iteration	1	2	3	4	5	6	7
node 1	🟧	⬜	⬜	⬜	⬜	🟧	🟧
node 2	🟥	🟧	🟧	🟧	🟥	🟥	🟥
node 3	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 4	🟥	🟥	🟥	🟥	🟥	🟥	🟥
node 5	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 6	⬜	⬜	⬜	🟧	🟥	🟥	🟧
node 7	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 8	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 9	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
node 11	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 12	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 13	⬜	⬜	⬜	🟧	🟧	🟧	🟥
node 14	⬜	🟧	🟧	🟧	🟧	🟧	🟧
node 15	⬜	⬜	⬜	⬜	⬜	⬜	⬜
node 16	🟥	🟥	🟧	⬜	⬜	⬜	⬜

Figure 4 7 iterations of the Probabilistic MPro on the Network Flow problem. Red cells have greater than 75 percent probability of being in top 7; orange greater than 50 percent. The true list is discovered after only 5 iterations. Notice, though that node 16 enters, but drops out after 3 iterations.

The Probabilistic Minimal Probing on the Network Flow problem finds the true list in only 5 iterations. Notice that there is more 'movement' in the list - for instance, node 16 enters the top 7 but exits by iteration 4. This is a feature of multivariate affine functions that determine the Network Flow and the estimation of dependency within location, across data.

References

1. Kobi Abayomi and Pablo Vanegas, *Bayesian multivariate markov processes for a network flow optimization problem*, Submitted to IEEE Transactions.
2. D. Suciu C. Re, N. Dalvi, *Efficient top-k query evaluation on probabilistic data*, ICDE 2007 (2007), 886 – 895.
3. F. Cao, K. Abayomi, and N. Gebraeel, *Probabilistic 'best set' sorting algorithms for multivariate (prognostic) data.*, Under Review,Ieee Transactions on Automation Science and Engineering (2013).
4. M. Soliman F. Ilyas, G. Beskales, *A survey of top-k query processing techniques in relational database systems*, ACM Computing Surveys **40** (2008), no. 4, 1–58.
5. R. Fagin, *Combining fuzzy information from multiple systems*, PODS 1996 (1996), 216–226.
6. N Gebraeel, *Prognostics-based identification of the top-k units in a fleet*, IEEE Transactions on Automation Science and Engineering **7** (2010), no. 1, 37–48.
7. G. Kollios D. Srivastava K. Yi, F. Li, *Efficient processing of top-k queries in uncertain databases with x-relations*, IEEE transactions on knowledge and data engineering **20** (2009), no. 12, 1669–1682.
8. S. Hwang KC. Chang, *Minimal probing: Supporting expensive predicates for topk queries*, SIGMOD 2002 (2002), 346 – 357.
9. KC. Chang M. Soliman, I. Ilyas, *Top-k query processing in uncertain databases*, ICDE 2007 (2007), 896 – 905.
10. R. Schenkel M. Theobald, G. Weikum, *Top-k query evaluation with probabilistic guarantees*, VLDB 2004 **30** (2004), 648–659.
11. W. Lee D. Lee M. Ye, X. Liu, *Probabilistic top-k query processing in distributed sensor networks*, ICDE 2010 (2010), 585–588.
12. Albert W. Marshall and Ingram Olkin, *Inequalities: Theory of majorization and its applications*, Academic Press, 1979.
13. A. Natsev, Y. Chang, J. Smith, C. Li, and J. Vitter, *Supporting incremental join queries on ranked inputs*, VLDB 2001 (2001), 281–290.
14. Roger Nelsen, *An introduction to copulas*, Springer, 2006.
15. S. Nepal and M. Ramakrishna, *Query processing issues in image(multimedia) databases*, ICDE 1999 (1999), 22–29.
16. A. Lote R. Fagin and M. Naor, *Optimal aggregation algorithms for middleware*, PODS 2001 (2001).
17. Stuart Russell and Peter Norvig, *Artificial intelligence, a modern approach*, Prentice Hall.
18. A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang, *A sampling-based approach to optimizing top-k queries in sensor networks*, ICDE 2006 (2006), 68–78.
19. Pablo Vanegas, Dirk Cattyrsse, and Jos Van Orshoven, *Allocating reforestation areas for sediment flow minimization: an integer programming formulation and a heuristic solution method.*
20. Y. Zhang J. Pei W. Wang W. Zhang, X. Lin, *Threshold-based probabilistic top-k dominating queries*, The VLDB Journal **19** (2010), no. 2, 283–305.
21. David Williams, *Probability with martingales*, Cambridge Mathematical Textbooks, Cambridge University Press, 1991.