JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 11, Number 5, 2004

Pp. 867–886

Efficient Exact *p*-Value Computation for Small Sample, Sparse, and Surprising Categorical Data

GILL BEJERANO,[1] NIR FRIEDMAN,[2] and NAFTALI TISHBY[2]

ABSTRACT

A major obstacle in applying various hypothesis testing procedures to datasets in bioinformatics is the computation of ensuing *p*-values. In this paper, we define a generic branch-and-bound approach to efficient exact *p*-value computation and enumerate the required conditions for successful application. Explicit procedures are developed for the entire Cressie–Read family of statistics, which includes the widely used Pearson and likelihood ratio statistics in a one-way frequency table goodness-of-fit test. This new formulation constitutes a first practical exact improvement over the exhaustive enumeration performed by existing statistical software. The general techniques we develop to exploit the convexity of many statistics are also shown to carry over to contingency table tests, suggesting that they are readily extendible to other tests and test statistics of interest. Our empirical results demonstrate a speed-up of orders of magnitude over the exhaustive computation, significantly extending the practical range for performing exact tests. We also show that the relative speed-up gain increases as the null hypothesis becomes sparser, that computation precision increases with increase in speed-up, and that computation time is very moderately affected by the magnitude of the computed *p*-value. These qualities make our algorithm especially appealing in the regimes of small samples, sparse null distributions, and rare events, compared to the alternative asymptotic approximations and Monte Carlo samplers. We discuss several established bioinformatics applications, where small sample size, small expected counts in one or more categories (sparseness), and very small *p*-values do occur. Our computational framework could be applied in these, and similar cases, to improve performance.

Key words: *p*-value, exact tests, branch and bound, real extension, categorical data.

1. INTRODUCTION

STATISTICAL NONPARAMETRIC TECHNIQUES ARE ROUTINELY USED in bioinformatics in the exploration of empirical data (Ewens and Grant, 2001). A common example is the many hypothesis rejection procedures, often resulting in the computation of *p*-values in a distribution free setting.

[1]Center for Biomolecular Science and Engineering, School of Engineering, University of California, Santa Cruz, CA 95064.
[2]School of Computer Science and Engineering, The Hebrew University, Jerusalem 91904, Israel.

In many one- or two-sided scenarios, given an outcome with an associated test statistic value t, we are required to sum the tail associated with the probability of all possible outcomes yielding a value of t or more. When we consider a finite sample space, we can directly compute the p-value by scanning all possible outcomes. However, in most real life problems, this direct approach is unfeasible as the number of possible outcomes is extremely large (typically exponential in the number of observations in our sample). Thus, in practice, one has to resort to using either asymptotic approximations or stochastic simulation methods.

With the proliferation of computing power, various algorithms have been devised to handle discrete datasets (Agresti, 1992, 2001). And while these different instances can be grouped by their underlying computational approach, not all such groups have been fully characterized.

In this paper, we define the generic algorithmic framework and preconditions behind the branch-and-bound approach to efficient computation of exact p-values and similar sums. Instead of explicitly enumerating all possible outcomes and for each one examining separately whether it passes the test threshold, we attempt to examine large groups of outcomes. If all outcomes in a group pass the test, or if all fail it, we can handle them without considering each individually. By careful design, these algorithms perform a systematic examination of possible groups, ensuring the computation of exact p-values. Many instances applying this approach to statistical computations already exist. Examples include Mehta and Patel (1983) and several related works discussed by Agresti (1992) and solutions to other problems such as those of Rue (2001), van de Wiel (2001), and Welch and Gutierrez (1988), and others.

Guided by the general framework, we introduce novel techniques and efficient exact algorithms for one- and two-way tables. We show empirically that this design indeed leads to a decrease in the computational complexity of performing the exact test, and greatly extends its range of feasibility. The resulting algorithms are shown to perform particularly well in contexts which abound in bioinformatic research—small samples, sparse null hypotheses, and rare events.

The paper is organized as follows. We begin with a case study. In Section 2, we define a goodness-of-fit test, Section 3 lists existing approaches to measure its p-value, and in Section 4 we develop and evaluate a novel algorithm for exact p-value computation of the likelihood ratio statistic in this context. Section 5 extends the algorithm to allow the use of any Cressie–Read statistic, including Pearson's X^2. In Section 6, we demonstrate how the same techniques can be applied to other hypothesis tests, such as two-dimensional contingency tables. Section 7 summarizes the results, potential bioinformatic applications, and future directions. Two technical appendices conclude the paper.

2. CASE STUDY: ONE WAY FREQUENCY TABLES

Let X be a discrete random variable with a finite set of possible values or categories $\{1, 2, \ldots, k\}$. Let Q be a multinomial distribution over this set, $Q = (q_1, q_2, \ldots, q_k)$. Assuming that X is governed by an unknown multinomial distribution Π_0, we want to decide whether

$$H_0 : \Pi_0 = Q$$
$$H_1 : \Pi_0 \neq Q \tag{1}$$

given a set of n independent observations of X. The composition of a column of multiply aligned biosequences yields one such scenario. Let $S_n = \{x_1, x_2, \ldots, x_n\}$ denote the sample. Let T_n denote its empirical *type* which counts how many times each possible value appeared in the sample, $T_n = (n_1, n_2, \ldots, n_k)$ where each $n_i = |\{j | x_j = i\}|$. Statistic T_n is a sufficient statistic of S_n. And let P_n denote its empirical probability distribution, $P_n = (p_1, p_2, \ldots, p_k) = \left(\frac{n_1}{n}, \frac{n_2}{n}, \ldots, \frac{n_k}{n}\right)$.

Generalizing the approach advocated by Radlow and Alf, Jr. (1975), we quantify the extent to which the observed type deviates from the null hypothesis distribution, using a chosen *test statistic* (or *discrepancy measure*). A test statistic D is a real valued function $D : \mathcal{T}_n \to R$, where

$$\mathcal{T}_n = \left\{ (n_1, n_2, \ldots, n_k) \,|\, \forall i : n_i \in N, \ \sum_{i=1}^{k} n_i = n \right\}$$

is the *sample space*, the collection of all possible empirical types of size n. Denote by $d_n = D(T_n)$ the value the statistic attains for a given sample. Two widely used discrepancy measures are the Pearson statistic

$$X^2 = \sum_{i=1}^{k} \frac{(n_i - nq_i)^2}{nq_i} = \sum_{i=1}^{k} \frac{n_i^2}{nq_i} - n$$

and the closely related likelihood ratio statistic

$$G^2 = -2 \log \frac{Q(S_n)}{P_n(S_n)} = 2nD_{KL}(P_n||Q) = 2 \sum_{i=1}^{k} n_i \log \frac{n_i}{nq_i} \tag{2}$$

where $D_{KL}(P||Q) = \sum_{i=1}^{k} p_i \log \frac{p_i}{q_i}$ is the Kullback–Leibler divergence (see, e.g., Lin, 1991). We can now define a hypothesis test for some chosen discrepancy measure D and a given sample:

Compute $d_n = D(T_n)$; reject H_0 iff $d_n \geq$ a predetermined threshold.

The p-value of this test is defined as the probability to draw under H_0 a sample of equal size for which the chosen discrepancy measure is at least as large as that of our observed sample,

$$p\text{-value} = Q(D(T_n') \geq d_n). \tag{3}$$

3. MEASURES OF p-VALUE

3.1. Exact tests

Denote the multinomial probability of drawing a sample with empirical type T_n, when $X \sim Q$, as

$$Q(T_n) = n! \prod_{i=1}^{k} \frac{q_i^{n_i}}{n_i!}. \tag{4}$$

A direct approach to evaluate (3) would explicitly sum

$$p\text{-value} = \sum_{\substack{T_n' \in \mathcal{T}_n \text{ s.t.} \\ D(T_n') \geq d_n}} Q(T_n') \tag{5}$$

by examining all possible types of size n. This approach is practicable only for small sample sizes over a small set of categories since the number of types examined $|\mathcal{T}_n| = \binom{n+k-1}{n}$ grows rapidly.

An alternative, theoretically exact approach computes the characteristic function of the chosen statistic and then inverts it using the fast Fourier transform (e.g., Baglivo *et al.*, 1992). The method is appealing as computation time increases only polynomially with sample size. However, its use of trigonometric functions and complex arithmetics introduces significant precision errors into the computation (Agresti, 1992).

Thus, both SAS (SAS 8, 1999) and StatXact (StatXact 5, 2001), two widely used software tools, perform the exhaustive enumeration defined above.

3.2. Asymptotic approximation

Perhaps the most common approach is to use statistics whose asymptotic distributions, for a wide range of values, have simple forms and are easily computable. For example, under broad regularity conditions, both X^2 and G^2 are distributed χ^2 in the limit, with $k - 1$ degrees of freedom (see Brownlee [1965] for proofs). Thus, for these and other so-called chi-square statistics, one uses

$$p\text{-value} \simeq \Pr(\chi_{k-1}^2 \geq d_n).$$

However, the convergence rate of the actual statistic to its limiting distribution is not always known and usually varies with the choice of parameters in H_0. Thus, when using the χ^2 approximation, we need to ensure that the asymptotic bound applies to the problem at hand (for references to the relevant voluminous literature, see Hutchinson [1979] and Read and Cressie [1988]). To account for *small sample* cases researchers have come up with several different correcting terms to the asymptotic expressions, trying to better approximate the exact value (Read and Cressie [1988] compare some of them). *Sparse null hypotheses*, or *sparse distributions*, are hypothesis testing instances where some of the expected counts are small, most commonly defined as a test where $nq_i \leq 5$ for at least one index i. These cases also fall outside the asymptotic χ^2 approximation assumptions. Unfortunately, both small sample size and sparse distribution settings are quite common in bioinformatics. Until recently, practitioners were urged to merge or ignore rare categories (see, e.g., Siegel and Castellan [1988] and Sokal and Rohlf [1995]) to circumvent this situation. However, merging schemes vary, yielding different p-values for the same dataset, as well as increasing the divergence of the obtained value from the exact p-value.

3.3. Simulation

With the advent of computing power, Monte Carlo methods using computer simulation have become widely practiced. A simple simulation approach to estimate (3), known as Monte Carlo integration, draws R (pseudo-random) i.i.d. samples, $\{S_n^{(1)}, \ldots, S_n^{(R)}\}$ from distribution Q and approximates

$$p\text{-value} \simeq \frac{|\{r \mid D(T_n^{(r)}) \geq d_n\}|}{R}.$$

As the variance of this estimator decreases as $\frac{1}{R}$, independent of the dimensionality of the sample space, it is very useful for computing p-values of the magnitude of the ubiquitous 0.05 value. However, smaller p-values and more accurate estimates require a great number of samples, of the order of $p\text{-value}^{-1}$. More sophisticated sampling schemes can be devised, in certain situations, to somewhat improve sampling accuracy, including Monte Carlo Rescue procedures (e.g., Senchaudhuri *et al.*, 1995) and successively refined Markov chain Monte Carlo sampling (Gilks *et al.*, 1996).

4. EFFICIENT EXACT p-VALUE COMPUTATION: THE G^2 EXAMPLE

4.1. Motivation

Consider a simple example. Let $k = 3$, $Q = (.1, .45, .45)$, $n = 2$, and $T_n = (1, 0, 1)$. One way to explicitly enumerate all possible types in order to perform the exact test (5) is through recursion, as illustrated in Fig. 1. We assign every possible value to n_1, for each we assign every possible value of n_2, and etc. At the leafs of the recursion tree, we find all six possible assignments. We can calculate G^2 for each and accumulate the mass of those who are at least as big as $d_n = G^2(T_n) \simeq 3.43$.

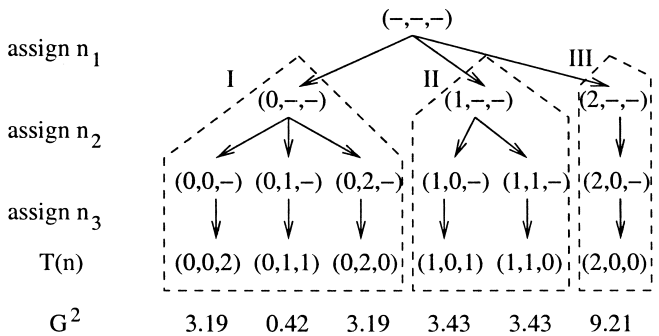

FIG. 1. Exact test recursion tree. We recursively develop all possible types for $k = 3$ and $n = 2$, by assigning all allowed values to each category in turn. Below the tree we write each type's G^2 statistic for $Q = (.1, .45, .45)$.

Note, however, that if we had tight upper and lower bounds on the values G^2 obtains in a given subtree, we could have ended our computation after assigning only n_1: The maximal G^2 value in subtree I, falls below d_n and thus this whole subtree can be discarded, while the minimal G^2 value in subtrees II, III are equal to or exceed d_n and the probability mass of all types they contain can be immediately accumulated. Thus, in retrospect, we could have examined only the top three nodes and concluded with the exact answer. We turn to formalize and extend these branch-and-bound intuitions.

4.2. Recursive domain partitioning

We define a *partial assignment* of a type of size n, denoted τ_n, as an assignment to a subset of the k variables $(n_1, \ldots, n_k)$ that can be completed to a valid empirical type of size n. In the example above, $\{n_1 = 0\}$ is a valid partial assignment. We write it succinctly as $\tau_n = (0, -, -)$, where "$-$" denotes a yet unassigned type. Formally, the set of all valid (strictly) partial assignments is

$$\mathcal{T}_n^{par} = \left\{ (n_1, n_2, \ldots, n_k) \mid \forall i : n_i \in \{-, 0, 1, \ldots, n\},\ \exists i : n_i = \text{`}-\text{'},\ \sum_{n_i \in N} n_i \leq n \right\}.$$

For a partial assignment τ_n, define $I = \{i | n_i \in N\}$ and $\overline{I} = \{i | n_i = \text{'}-\text{'}\}$ as the sets of assigned, and yet unassigned categories, respectively, and let

$$\overline{n} = n - \sum_{i \in I} n_i, \quad \overline{q} = 1 - \sum_{i \in I} q_i = \sum_{i \in \overline{I}} q_i, \quad \overline{q}_{min} = \min_{i \in \overline{I}} \{q_i\}.$$

In our example, for $\tau_n = (0, -, -)$: $I = \{1\}$, $\overline{I} = \{2, 3\}$, $\overline{n} = 2$, $\overline{q} = .9$, and $\overline{q}_{min} = .45$.

Let $[\tau_n]$ be the set of all empirical types which can be completed from τ_n,

$$[\tau_n] = \{(n'_1, \ldots, n'_k) \in \mathcal{T}_n \mid \forall i \in I(\tau_n),\ n'_i = n_i\}.$$

We define the probability of τ_n, under the null hypothesis, as the sum of the probabilities of all types in $[\tau_n]$,

$$Q(\tau_n) = \sum_{T_n \in [\tau_n]} Q(T_n) = n! \frac{\overline{q}^{\overline{n}}}{\overline{n}!} \prod_{i \in I} \frac{q_i^{n_i}}{n_i!}. \tag{6}$$

The latter equality can be directly derived as a special case of (4), where each category $i \in I$ is assigned the count n_i with probability q_i, leaving a total sum of $\overline{n}$ to be assigned to an "elsewhere" bin, with (cumulative) probability $\overline{q}$. For $\tau_n = (0, -, -)$: $[\tau_n] = \{(0, 0, 2), (0, 1, 1), (0, 2, 0)\}$, and $Q(\tau_n) = .81$.

We define a *recursion tree* as a structured way to recursively enumerate the set of empirical types $\mathcal{T}_n$: Let σ be a permutation of size k. The tree that matches σ, denoted $\mathcal{A}_\sigma$, is a tree where the root node contains the empty assignment $(-, \ldots, -)$. Extend from it all allowed assignments to category $n_{\sigma(1)}$. From each of these, extend all allowed assignments to category $n_{\sigma(2)}$, etc. In Fig. 1 we have the recursion tree for $k = 3$, $n = 2$ that matches the identity permutation. Using the appropriate permutation, we could have defined a different recursion tree that first assigned n_3, then n_2, and finally n_1. Note that any such tree has a uniform depth k, and its set of leafs is exactly $\mathcal{T}_n$. Moreover, the set of leafs in a subtree rooted at any τ_n is exactly the set $[\tau_n]$, and for every inner node, the set $[\tau_n]$ is a disjoint union of the sets of types held in its sons.

4.3. Bounding the statistic

Having defined how to recursively partition the summation domain, we move to bound the statistic on subdomains, by defining

$$G^2_{max}(\tau_n) = \max_{T_n \in [\tau_n]} G^2(T_n), \quad \text{and}$$

$$G^2_{min}(\tau_n) = \min_{T_n \in [\tau_n]} G^2(T_n).$$

Lemma 1. *For any* $\tau_n \in \mathcal{T}_n^{par}$,

$$G^2_{max}(\tau_n) = 2\left(\sum_{i\in I} n_i \log\frac{n_i}{nq_i} + \overline{n}\log\frac{\overline{n}}{n\overline{q}_{min}}\right), \quad \textit{and} \tag{7}$$

$$G^2_{min}(\tau_n) \geq 2\left(\sum_{i\in I} n_i \log\frac{n_i}{nq_i} + \overline{n}\log\frac{\overline{n}}{n\overline{q}}\right). \tag{8}$$

Proof. Let $\overline{I}$ denote the indices of the yet unassigned categories of τ_n. Consider the *real extension* of G^2 over the set of all nonnegative real types that sum to n. Differentiating G^2 with respect to the unassigned counts n_i, we get that the Hessian of G^2 is a diagonal matrix,

$$\forall i, j \in \overline{I}: \quad \left[\frac{\partial^2 G^2}{\partial n_i \partial n_j}\right] = \delta_{ij}\frac{2}{n_i}$$

where δ_{ij} is Kronecker's delta function. Since $\forall i : n_i \geq 0$, the Hessian is positive definite, and we conclude that G^2 is strictly convex over its domain (Rockafellar, 1970, p. 27).

To find the minima of G^2, we use Lagrange multipliers. We define the Lagrangian

$$J = 2\sum_{i=1}^{k} n_i \log\frac{n_i}{nq_i} - \gamma\left(\sum_{i\in\overline{I}} n_i - \overline{n}\right).$$

By solving $\nabla J = 0$, we obtain the solution $\forall i \in \overline{I} : n_i = \frac{q_i}{\overline{q}}\overline{n}$. Since G^2 is strictly convex, this interior point must be a global minimum (Rockafellar, 1970, p. 242). In general, this will not yield a valid integer assignment, but it does bound G^2_{min} from below, obtaining (8).

Since G^2 is convex, it achieves its maximum value in extreme points of the allowable region (Rockafellar, 1970, p. 343), that is, on the vertices of the set of possible assignments. Recall that the vertices are the assignments where all the remaining counts are assigned to one category. Now, let $l \in \overline{I}$ attain the least yet unassigned probability, $q_l = \overline{q}_{min}$. Clearly,

$$\forall i \in \overline{I}: \quad \log\frac{\overline{n}}{nq_l} \geq \log\frac{\overline{n}}{nq_i}.$$

Thus, assigning all $\overline{n}$ remaining counts to n_l maximizes G^2 over $[\tau_n]$ and yields (7). ■

Note that G^2_{max} was achieved by assigning all remaining counts, $\overline{n}$, to the least probable remaining category $\overline{q}_{min}$, and that G^2_{min} was bounded from below using an assignment which attains the minimal possible value over the set of all not-negative *real* assignments that sum to $\overline{n}$. In general it will involve fractional counts, and the minimal allowed assignment in integers will be somewhat higher. To simplify notations, we will next use (8) as the value of G^2_{min} with the understanding that it can be replaced by a tighter bound or indeed by the exact minimum, when either is easy to obtain.

4.4. The algorithm

We can now utilize the domain partitioning as the "branch" step of the algorithm alluded to in Section 4.1. The easily computable bounds on the statistic in a given partition and the probability measure of the partition comprise the "bound" step. The resulting algorithm is given in Fig. 2. Proof of correctness is immediate and thus omitted.

This algorithm always returns the exact p-value. Furthermore, the faster it runs compared to the exhaustive test, the more superior it is in terms of arithmetic precision. This is true because adding a big subtree as a single partial assignment is much more accurate than summing the diminishing probabilities of each type within it. Note also that the p-value of a given test depends on the observed sample S_n only through the magnitude of its statistic d_n. The above algorithm can thus be easily altered to simultaneously

Call format:

p-value = descend$(1, (-, \ldots, -))$

descend$(i, (n_1, \ldots, n_k))$

```
{  p-val = 0
   for n_i = 0 to n − Σ_{j=1}^{i−1} n_j
   {  τ_n = (n_1, ..., n_i, −, ..., −)
      if G²_min(τ_n) ≥ d_n
         p-val = p-val + Q(τ_n)
      else if G²_max(τ_n) ≥ d_n
         p-val = p-val + descend(i + 1, τ_n)
   }
   return(p-val)
}
```

Description:

Pick an order σ in which to assign the categories (optimized in Sec. 4.6). Start traversing the recursion tree $\mathcal{A}_\sigma$ from the empty assignment root node. For each node, or partial assignment τ_n:

Pruning criterion 1 *If $G^2_{min}(\tau_n) \geq d_n$, add $Q(\tau_n)$ to the accumulating probability measure, and do not descend into the sub-tree rooted in τ_n.*

Pruning criterion 2 *If $G^2_{max}(\tau_n) < d_n$, discard the sub-tree rooted in τ_n, and do not descend further.*

If neither criterion holds, descend into all sons of τ_n (all allowed partial assignments to the next category), and examine each, recursively.

FIG. 2. Efficient exact p-value computation for the likelihood ratio goodness-of-fit test. For ease of exposition, we assume σ to be the identity permutation in the pseudo-code on the left and assign first n_1, then n_2, etc.

retrieve the p-values of several observed samples of the same size in a single traversal, by tracking in each step only those values which are yet to be resolved. This is an easy starting point to critical value computation and look-up table generation in a single traversal. Another simple alteration can yield mid-p values, defined as half the probability of the observed result d_n plus the probability of the more extreme values. This value is sometimes used to eliminate problems arising from discreteness (Agresti, 1992).

4.5. A faster convex variant

Consider the basic step in Fig. 2, which iterates over all allowed values of n_i, the assignment to the *next* category. If the changes in G^2_{max} and G^2_{min} as a function of n_i have simple mathematical forms, we could handle groups of n_i values without examining each separately.

Let τ_n be a partial type at some level $i-1$, which needs to be descended (i.e., $G^2_{min}(\tau_n) < d_n \leq G^2_{max}(\tau_n)$). Denote, with a slight abuse of notation, $\overline{n} = n - \sum_{j=1}^{i-1} n_j$, $\overline{q} = \sum_{j=i+1}^{k} q_j$ and $\overline{q}_{min} = \min_{\{j=i+1,\ldots,k\}} q_j$. We need to assign to the next category n_i all possible values $\{0, 1, \ldots, \overline{n}\}$ and examine G^2_{max}, G^2_{min} for each. Since both bounds (7), (8) have the same form as a function of n_i, we can write compactly

$$G^2_{bound}(n_i) = 2\left(\sum_{j=1}^{i-1} n_j \log\frac{n_j}{nq_j} + n_i \log\frac{n_i}{nq_i} + (\overline{n} - n_i)\log\frac{\overline{n} - n_i}{n\overline{q}_\star}\right)$$

where $G^2_{bound} = G^2_{max}$ for $\overline{q}_\star = \overline{q}_{min}$ and $G^2_{bound} = G^2_{min}$ for $\overline{q}_\star = \overline{q}$. It is easy to verify that G^2_{bound} is strictly convex in n_i when $n_i \in [0, \overline{n}]$ and obtains its minimum at $n_\star = \frac{q_i}{q_i + \overline{q}_\star}\overline{n}$ (which is in general fractional). Since by definition $\overline{q}_{min} \leq \overline{q}$, the "swoosh"-like shape of the two bounds as a function of n_i is as shown in Fig. 3 (whereas the vertical ordering of points $G^2_{max}(n_i = \overline{n}) = G^2_{min}(n_i = \overline{n})$, $G^2_{max}(n_i = 0)$,

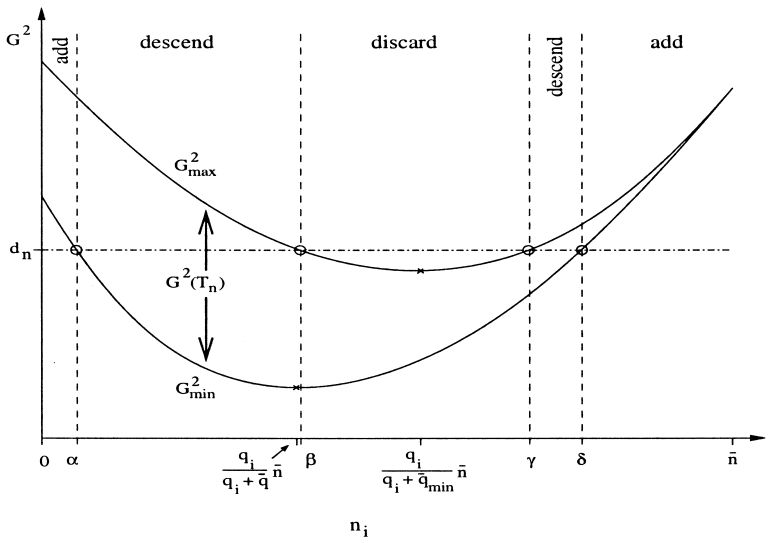

FIG. 3. Faster node computation for a convex statistic. We plot the values of G^2_{max} and G^2_{min} versus all possible real assignments to the next category n_i. A threshold d_n can intersect each of the two convex curves at most twice. The four intersection values, denoted $\alpha, \beta, \gamma, \delta$, define five groups of integer n_i values. All values in each group are equally treated: values between $\{0, \ldots, \lfloor\alpha\rfloor\}$ and $\{\lceil\delta\rceil, \ldots, \overline{n}\}$ are added to the accumulating p-value (pruning criterion 1), values between $\{\lceil\beta\rceil, \ldots, \lfloor\gamma\rfloor\}$ are discarded (pruning criterion 2), and the rest need to be further descended (see Fig. 2).

and $G^2_{min}(n_i = 0)$ is inverse to that of q_i, $\overline{q}_{min}$ and $\overline{q}$, respectively). Clearly, any threshold d_n can intersect either curve at most twice. The four (or less) intersection values, denoted $\alpha - \delta$ in Fig. 3, define five (or fewer) groups of n_i values. All values in each group are equally treated. Based on our analysis of G^2_{bound}, we can now perform four *binary searches* (Cormen *et al.*, 1990), to elucidate $\lfloor\alpha\rfloor$, $\lceil\beta\rceil$, $\lfloor\gamma\rfloor$, and $\lceil\delta\rceil$, by searching for d_n with G^2_{max} and G^2_{min} over their respective $\{0, \ldots, \lfloor n_\star\rfloor\}$ and $\{\lfloor n_\star\rfloor + 1, \ldots, \overline{n}\}$, which are all sorted.

By identifying the groups of n_i values that require equal treatment, we save the cost of evaluating the bounds for each possible choice of n_i. While the algorithm of Fig. 2 computed about $2\overline{n}$ bounds, we now perform only about $2 \log \overline{n}$ computations at every node.

One further improvement is made easy within the Convex procedure. Reconsider Fig. 3. If more than half the types need to be added to the accumulating p-value mass, we can instead add the probability mass of the father node and subtract those of the complementary set of types to achieve the same increment using fewer mathematical operations.

To conclude, note that this variant performs exactly the same number of recursive calls as the algorithm in Fig. 2. However, it reduces the amount of time spent in each invocation.

4.6. Computational complexity: An empirical study

We turn to evaluate the performance of the alternative algorithms described above in terms of runtime and precision of computation (for the approximation methods). In general, the cost of computing p-values depends on k, Q, n, and d_n. Our pruning algorithms also depend on the assignment order σ. In this section, we examine how each of these factors affects the cost of p-value computation. For this purpose, we implemented five computational procedures.

- Direct: exact computation by full recursive enumeration of all types, as described in Section 3.1. This procedure is equivalent to the ones implemented by SAS and StatXact.
- Pruned: exact computation by recursive enumeration with the two pruning criteria of Section 4.4.
- Convex: same as Pruned but exploits the convexity of the bounds on G^2, as in Section 4.5.
- χ^2: the chi-squared approximation discussed in Section 3.2.
- Simulation: the Monte Carlo sampler approximation from Section 3.3.

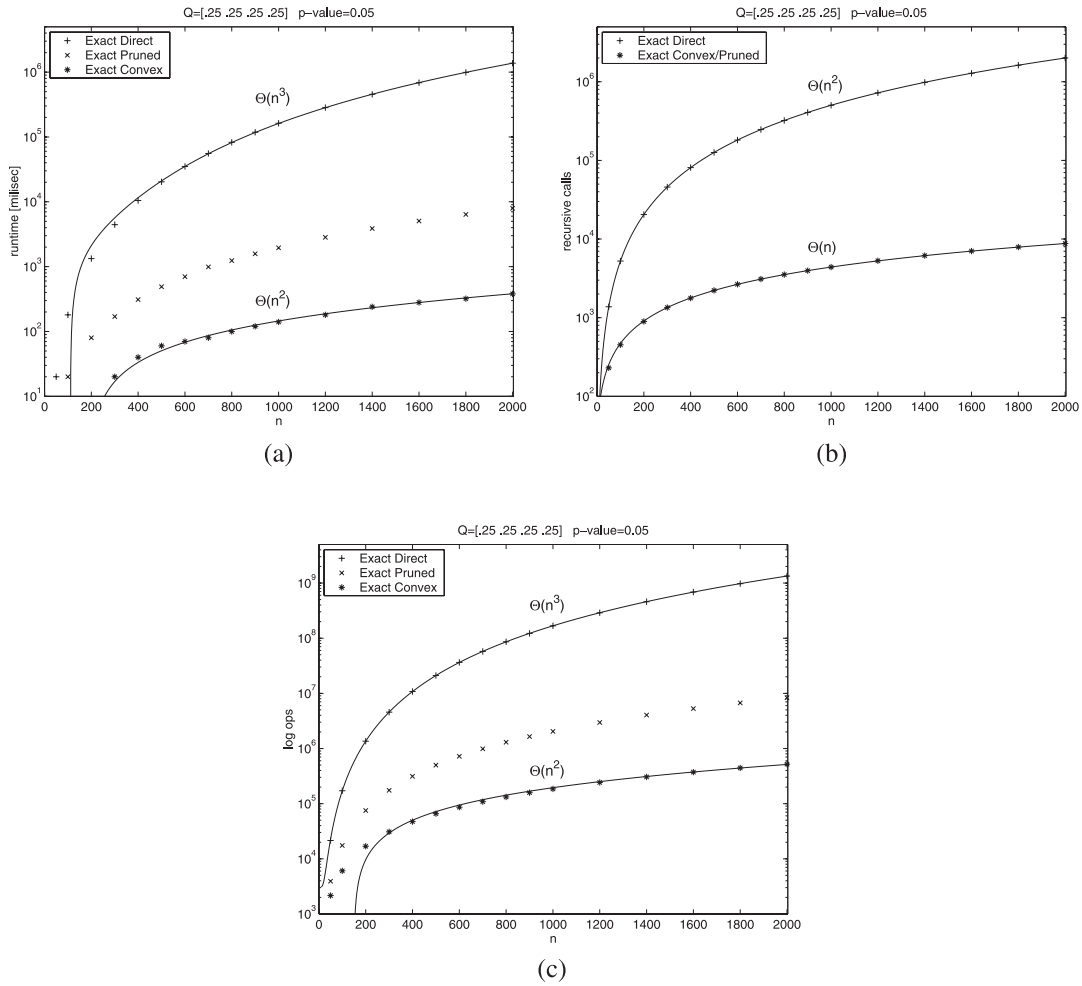

FIG. 4. Evaluation of the performance of our algorithms compared to direct computation of p-values. All three graphs plot the cost of p-value computation for $Q = (.25, .25, .25, .25)$ with different choices of n. The x-axes denote the number of samples, n. The y-axis denotes the computation cost, using three different performance measures: (**a**) runtime, (**b**) number of recursive calls, and (**c**) number of additions. For each n, the choice of d_n is set such that the resulting p-value is 0.05. Polynomials of the lowest acceptable degree are fitted against our measurements, and their degree is noted. Sublinear complexity in the sample space is evident even for this uniform Q which we later show to be our worst case in terms of performance.

All procedures employ further practical speed-ups which are detailed in Appendix B. They have been implemented in C and run on a Pentium III 733 MHZ Linux machine. The chi-squared distribution was computed as by Press *et al.* (1993). Time measurements were performed with the clock() function (Harbison and Steele, Jr., 1995). The code is available from the first author.

We start by comparing the performance of the first three, exact, algorithms on a simple problem. We performed a series of tests with a growing sample size for a uniform Q over $k = 4$ categories. All tests are set to result in the same p-value of 0.05, a widely used rejection boundary. Figure 4(a) shows the runtime of the three algorithms as a function of the sample size n.

For fixed k and growing n, $|\mathcal{T}_n| \simeq \frac{n^{k-1}}{(k-1)!}$. Indeed, this is made evident by fitting a cubic polynomial in n to the runtime measurements of Direct. On the other hand, the runtimes of Convex match a quadratic growth function.

To understand the source for this improvement, we considered two quantities that play a role in the runtime of these procedures. The first is the number of recursive calls made by each procedure. As we can see in Fig. 4(b), the number of recursive calls made by Convex and Pruned is again in perfect match

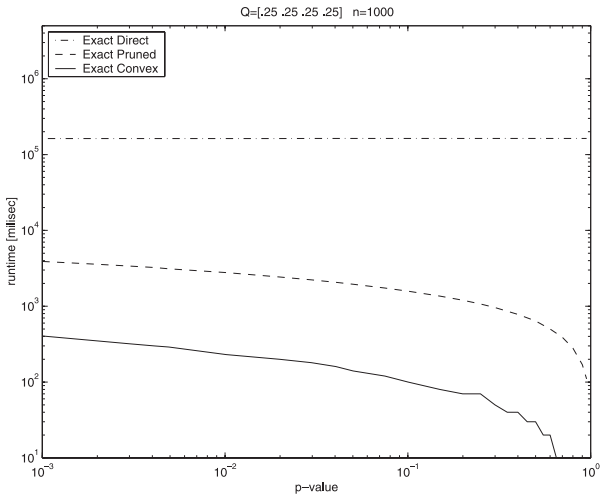

FIG. 5. The runtime cost (y-axis) of p-value computation as a function of the p-value of the threshold (x-axis). These were computed for uniform Q over four categories and $n = 1{,}000$.

with a polynomial of a lower degree than that of Direct. This indicates that recursive exploration of large subtress is avoided by the two pruning criteria.

The second quantity is the number of partial types added (and subtracted, in Convex) in accumulating the exact p-value. We concentrate on these operations as they are costly computationally. In order to preserve precision, our procedure maintains not probabilities but their logarithm. This makes multiplication fast, but additions and subtractions of probabilities are slow since they require computing the exponents of the arguments and then taking the logarithm of the result. In practice, one of the exponentiations can be saved (see Appendix B). In Fig. 4(c), we plot the number of such operations made by the three different procedures. Not only is the fitted degree of the polynomial for Convex lower than that of Direct, but we also see here the reason of the runtime improvement from Pruned to Convex in Fig. 4(a).

Interestingly, when the evaluation is repeated for a uniform distribution over five categories (not shown), the fitted degree of all polynomials increases by one, except the runtime and number of addition operations for Convex, which appears to be already well fitted by quadratic polynomials. The theoretical dependence between these sizes is under further investigation.

In the evaluation above, the p-value was kept fixed. Next, we examine the effect of the computed p-value on the runtime. As we can see in Fig. 5, when using Direct, the actual value of the result has little effect. As the p-value increases (for otherwise fixed conditions) there are more terms to sum. However, since the drop in type probability is very drastic away from Q, even for low p-values, most types need to be summed. As a result, the runtime of Direct in the chosen range is almost constant. For example, in Fig. 5, already at p-value 10^{-3} we sum 99.67% of the types. In the Pruned and Convex algorithms, the opposite happens—the bigger the p-value, the less work needs to be done—fewer recursive calls and fewer arithmetic operations (not shown) result in faster runtime.

When considering a nonuniform null-hypothesis Q, the permutation order indeed affects computation time. In Pruned, it has a significant, yet at times opposite, effect on the number of recursive calls and arithmetic operations, and thus a mixed-effect on runtime. However, in Convex, for many examined scenarios, the change in both recursive calls and arithmetic operations grows and drops together, and the best expansion order is always from smallest q_i to the largest one. The slowest order is always the reverse, but the net effect is usually small because of a small overall change in the number of log ops, as demonstrated in Fig. 6. From here on, we use Convex with smallest to largest q_i expansion.

Another factor that affects runtime is the *entropy* of Q. Recall that the entropy is defined as $H(Q) = -\sum_{i=1}^{k} q_i \log q_i$, and it measures the "information content" of the distribution. We have drawn 2,000 distributions uniformly from the space of all four categories distributions. Using these, we demonstrate in Fig. 7 the positive correlation between the entropy of Q and our runtime on it. Moreover, the sparser (Section 3.2) the distribution is among equal entropy Q's, the better the speed-up (e.g., compare points (c) and (d) in the figure).

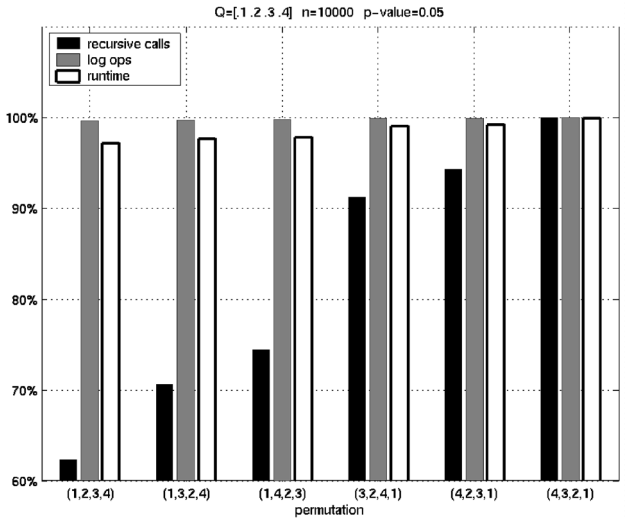

FIG. 6. The cost (recursive calls, arithmetic operations, and runtime) of p-value computation for different expansion orders in the Convex algorithm. All y-axis values increase monotonically, and each column is normalized by the result of the rightmost (slowest) permutation. The leftmost, fastest, permutation is seen to expand n in an ascending q_i order, for $Q = (.1, .2, .3, .4)$.

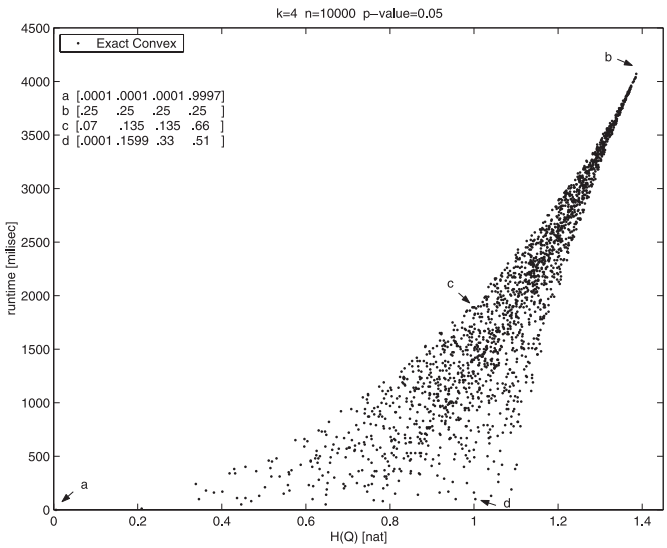

FIG. 7. The correlation between the entropy of Q (x-axis) and the runtime of computing p-values (y-axis). Each point corresponds to a choice of Q. Four points are labeled: (**a**), (**b**) at the two extremes of distribution entropy values and (**c**), (**d**) that have equal entropies and serve to demonstrate the speed-up effect caused by the sparseness of (**d**). Extrapolating from Fig. 4, we note that even our slowest result, computing point (**b**) in 4 seconds using Convex, would require about 55 hours using Direct.

The strong positive correlation between the sparseness of the underlying distribution and the performance boost is compelling when considering that the χ^2 approximation, which is computationally faster and usually quite accurate, is problematic in this regime. One way of explaining it ties to our results above. Typically, most types need to be summed (see discussion of Fig. 5). The value of G^2 on all these exceeds d_n. By expanding lower q_i values first, we add bigger $p_i log \frac{p_i}{q_i}$ terms first into the accumulating G^2 value and cross the d_n threshold earlier on (consider Fig. 6). This speedup becomes all the more evident the sparser Q is.

As the number of categories k in a test increases, so does the sample space, and with it, inevitably, our runtime. Figure 8 demonstrates this effect by comparing for a growing k our best runtime for a sparse

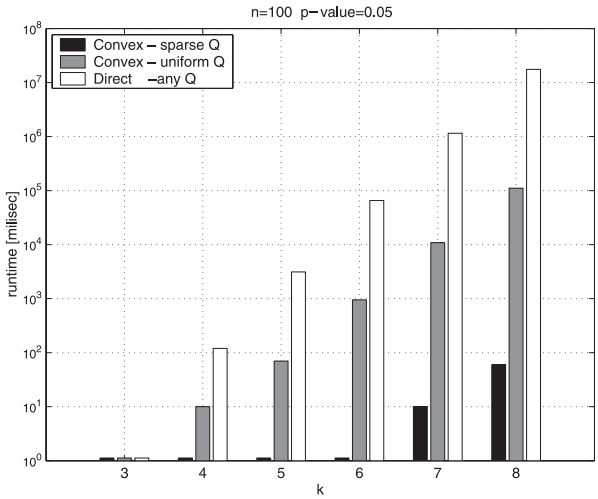

FIG. 8. Effect of the number of categories (x-axis) on the runtime of the algorithm (y-axis). The figure compares the increase in runtime for Direct and Convex when computing a p-value of 0.05 with $n = 100$. Convex computation time ranges between that for sparse Q's (here $n \cdot q_i = 1$ for all $i \neq k$) and for a uniform Q. For Direct all Q's require about the same computation time.

$Q = (\frac{1}{n}, \ldots, \frac{1}{n}, \frac{n-k+1}{n})$, our worst when Q is uniform, and the runtime of Direct, which is about the same for all Q's.

As computational resources constantly improve and proliferate, researchers who may spend months collecting data can now easily perform more computations in order to obtain satisfactory p-values. We give three examples of cases where our method outperforms the standard approximations. In Fig. 9(a), we plot the approximation error of χ^2 for a sparse (bump) distribution. Convex computation for this case is instantaneous. In Fig. 9(b), we plot the approximation error of the Monte Carlo sampler Simulation on a sharper (dip) distribution, which is sparse in a single coordinate and uniform elsewhere. By running Simulation three times for each value, we show that the variance of its estimate remains high, even when it is allowed four times more runtime than the slowest Convex exact computation in this setting. In Fig. 9(c), we exemplify the computation of extremely small p-values. As technology advances, several empirical sciences, notably molecular biology and neuroscience, generate ever increasing amounts of raw data. Therefore, one often scans through many different test combinations, searching for significant patterns in such huge datasets. The use of Bonferroni's or a similar correcting factor to compensate for the multiple tests results in a search for very small p-values, of the magnitude of our example. While the runtime of the exact Convex rises slowly with the decline of the p-value (observed already in Fig. 5), the Monte Carlo Simulation must now sample roughly inversely proportional to the p-value it tries to measure. Our simple sampler quickly becomes impractical in these settings, and as the required p-value decreases, so will the more sophisticated sampling methods, mentioned in Section 3.3.

To summarize, we have shown that for a broad range of n with a modest k, Convex allows one to perform the exact G^2 test. Its computational complexity was shown to be sublinear even in the worst case scenario of a uniform Q. It was also shown to be extremely fast and thus appealing for sparse distributions where the χ^2 approximation can be problematic. In parts of this region, it also outperforms Monte Carlo simulations in the sense that when the latter are allotted a comparable amount of runtime, the resulting approximation variance is high.

5. THE CRESSIE–READ STATISTICS

We now show that our algorithm for efficient exact p-value computation in fact maps to the whole family of Cressie–Read statistics. The Cressie–Read (or power-divergence) goodness of fit statistics are a single parameter family of discrepancy statistics. It is defined as follows.

$$D^{\lambda}(T_n) = 2nI^{\lambda}(P_n||Q) \qquad \forall \lambda \in R \tag{9}$$

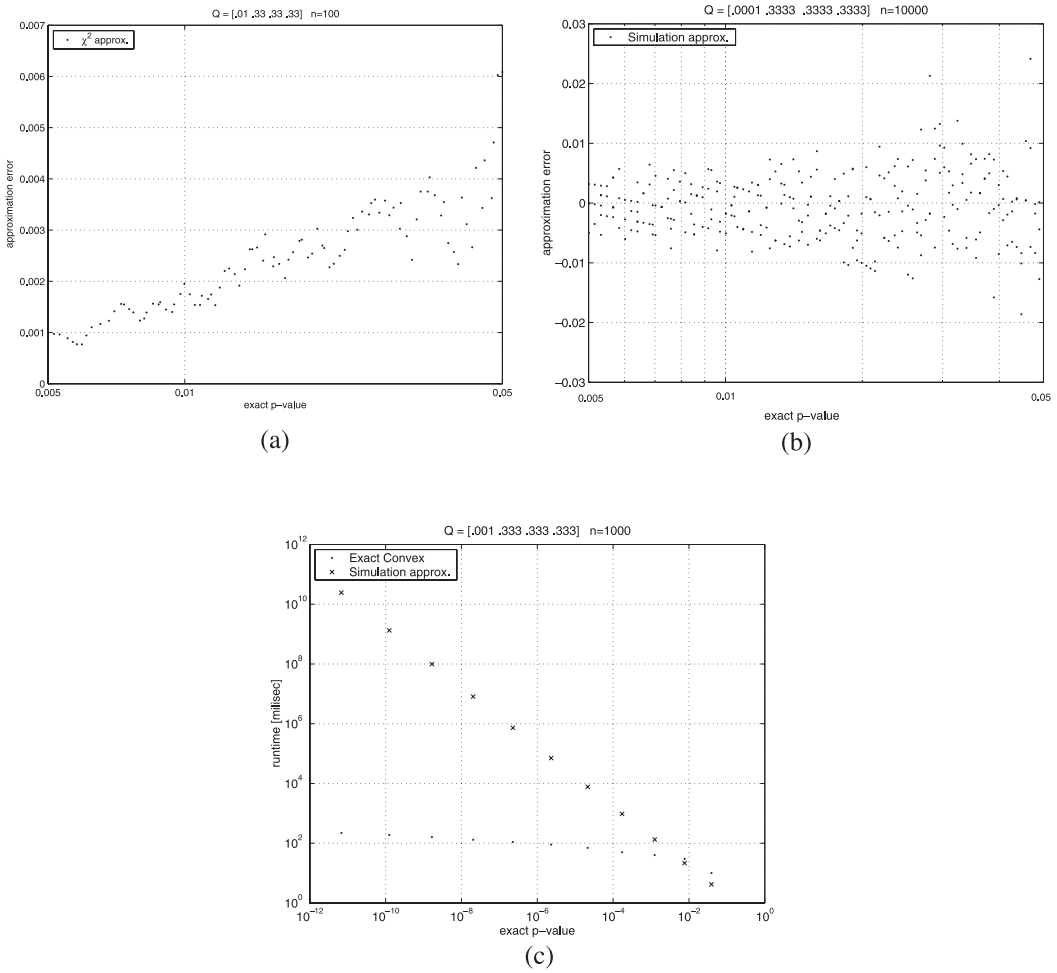

FIG. 9. Examples where common approximations are inferior to our method. (**a**) The inadequacy of the χ^2 approximation for a sparse "dip" distribution $Q = (.01, .33, .33, .33)$ with $n = 100$. We plot the approximation error (y-axis), which is the difference between the χ^2 value and the exact value, against various exact p-values (x-axis). Convex computation here is instantaneous for all p-values. (**b**) An example where the Monte Carlo sampler Simulation performs poorly compared to Convex on a sparse "dip" distribution $Q = (0.0001, 0.3333, 0.3333, 0.3333)$ and $n =$ 10,000. Using the same axes, we plot the Simulation approximation error when each run was allotted four times the runtime of the slowest Convex computation in this graph. For each exact value, we run Simulation three times to demonstrate the large estimation variance in the given time. (**c**) For $Q = (0.001, 0.333, 0.333, 0.333)$, we plot the exact p-value of the types $(i, \frac{1000-i}{3}, \frac{1000-i}{3}, \frac{1000-i}{3})$ for $i = 5, \ldots, 15$, right to left (x-axis). For each of these, we plot the actual runtime of Convex, and an extrapolation of the amount of time it would take Simulation to draw $p\text{-value}^{-1}$ samples in order to achieve an acceptable approximate value (y-axis).

where

$$I^{\lambda}(P_n||Q) = \begin{cases} \dfrac{1}{\lambda(\lambda+1)} \displaystyle\sum_{i=1}^{k} p_i \left[\left(\frac{p_i}{q_i} \right)^{\lambda} - 1 \right] & \forall \lambda \neq 0, -1 \\ D_{KL}(P_n||Q) = \lim\limits_{\lambda \to 0} I^{\lambda}(P_n||Q) & \lambda = 0 \\ D_{KL}(Q||P_n) = \lim\limits_{\lambda \to -1} I^{\lambda}(P_n||Q) & \lambda = -1 \end{cases}$$

If P_n is not strictly positive with respect to Q ($\exists i : q_i > p_i = 0$) for $\lambda \leq -1$, or vice versa for $\lambda > -1$, set $I^{\lambda} = \infty$. For all choices of λ, $I^{\lambda}(P||Q) \geq 0$, with equality iff $P = Q$. All D^{λ}'s are asymptotically distributed χ^2_{k-1} under H_0 with the usual regularity assumptions, allowing for the standard p-value

Table 1. Cressie–Read Instances Corresponding to Well-Known Statistics

Value	*Statistic*	*Symbol*
$\lambda = 1$	Pearson	X^2
$\lambda = 0$	Log likelihood ratio	G^2
$\lambda = -\frac{1}{2}$	Freeman–Tukey	T^2
$\lambda = -1$	Modified G^2	GM^2
$\lambda = -2$	Neyman modified X^2	NM^2

approximation (see Section 3.2). Moreover, this parametric family includes five well-known statistics, listed in Table 1. By extending these five to a continuous family, one can search for new statistics that combine desired features from the known ones. One can also better match different statistics from this wide spectrum to different scenarios. For details and proofs, see Read and Cressie (1988).

In order to map our pruning algorithm to this parametric family, we define as in Section 4.3

$$D^{\lambda}_{max}(\tau_n) = \max_{T_n \in [\tau_n]} D^{\lambda}(T_n), \quad \text{and}$$

$$D^{\lambda}_{min}(\tau_n) = \min_{T_n \in [\tau_n]} D^{\lambda}(T_n).$$

It is our intention to use the sample space decomposition of Section 4.4, replacing G^2_{max} and G^2_{min} (which are specific to $\lambda = 0$) with the above two functions.

Lemma 2. *For $\lambda > -1$, let Q be some null hypothesis, and let $\tau_n \in \mathcal{T}_n^{par}$ be a valid partial assignment. Denote by $l = \arg\min_{i \in \overline{I}} q_i$. Then,*

a. $D^{\lambda}_{max}(\tau_n)$ is obtained by assigning $n_l = \overline{n}$ and zero elsewhere, and $D^{\lambda}_{min}(\tau_n)$ is bounded from below by the assignment $\forall i \in \overline{I}: \quad n_i = \frac{q_i}{q}\overline{n}$.

b. D^{λ}_{max} and D^{λ}_{min} are both convex as functions of the next assignment variable n_i.

c. Cases where $\lambda \leq -1$ are also computable efficiently using similar arguments.

The proof extends the results of the previous section and is deferred to Appendix A. We have thus shown that all Cressie–Read statistics yield to the same efficient computational procedure as G^2 does, including the further convexity speedup of Section 4.5. Indeed, we have implemented our method for the Cressie–Read statistics and repeated the experiments of Section 4.6 using various $\lambda > -1$, including Pearson's X^2 and $\lambda = \frac{2}{3}$, which is advocated in Read and Cressie (1988). The behavior of the algorithm and magnitude of speed-up carried over in all cases (not shown).

6. EXTENSION TO $r \times c$ CONTINGENCY TABLES

Clearly, the branch and bound approach employed in the previous sections is quite general. It requires a recursive partitioning scheme of the sample space, coupled with the ability to efficiently compute the probability of a partial type, as well as tight lower and upper bounds on the value of the chosen statistic. To demonstrate the possible range of applications, we discuss another case of wide interest in statistics. Exact inference in two-way contingency tables has been researched extensively (e.g., reviewed by Agresti [1992] and Mehta and Patel [1997]). We briefly define the mathematical setting. The sample contains n observations of a pair of categorical variables, $x_m \in \{1, \ldots, r\} \times \{1, \ldots, c\}$. The type $T_n = (n_{11}, \ldots, n_{rc})$ holds the number of times each pair (i, j) appeared in the sample. Define the row and column margins of T_n as $r_i = \sum_j n_{ij}$ and $c_j = \sum_i n_{ij}$, respectively. Devise an independent joint probability distribution of the two variables from their margins $Q = (\frac{r_1 c_1}{n^2}, \ldots, \frac{r_r c_c}{n^2})$. The null hypothesis states that the sample was drawn i.i.d. according to Q. We return to the G^2 discrepancy measure. In the exact conditional approach,

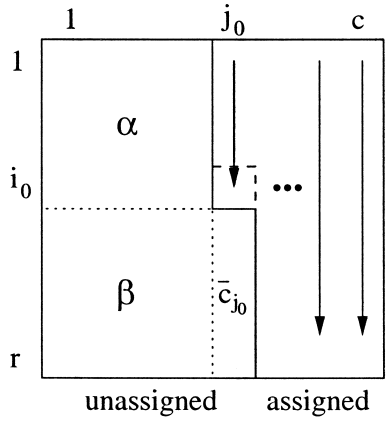

FIG. 10. Assignment order for a two-way contingency table recursion tree, column-wise starting top-right. The figure depicts a partial assignment $\tau_{i_0 j_0}$ up to some $n_{i_0 j_0}$, which respects the predetermined row and column margins. The division of the yet unassigned variables into three regions is used in the proof of Lemma 3.

the sample space with respect to $G^2(T_n)$ is the set of all possible (nonnegative integer) tables with the same row and column margins. The merits of this space are discussed by Agresti (1992) and by Mehta and Patel (1997).

The prominent approach to exact p-value computation in this scenario stems from the seminal work of Mehta and Patel (1983). We shall first define a depth-first tree approach to the problem following the guidelines and techniques developed herein. Then we will equate it with existing network variants. We begin by setting the order of assignment to $n_{1c}, n_{2c}, \ldots, n_{r-1c}, n_{1c-1}, n_{2c-1}, \ldots, n_{r-12}$, depicted in Fig. 10. Assignments to the last row and first column are determined by the margins, as we go along. Contrary to the network approach, we show that we are able to compute all three quantities of interest (probability and both bounds on the statistic) for an allowed partial assignment up to any $n_{i_0 j_0}$.

Lemma 3. *Denote the partial assignment above as $\tau_{i_0 j_0}$, then*

a. $Q(\tau_{i_0 j_0})$ is exactly computable as a simple product of hypergeometric probabilities;
b. $G^2_{min}(\tau_{i_0 j_0})$ is bounded by the minimal attainable value over all nonnegative real *assignments to the remaining variables, and the latter is easily computable in analytical form;*
c. $G^2_{max}(\tau_{i_0 j_0})$ is bounded by the two maximal attainable values in integer assignments, respecting only row or column margins alternatively (these two are easily computable in analytical form); and
d. both bounds allow for the efficient binary search approach of Section 4.5, as a function of the last assignment variable $n_{i_0 j_0}$.

The proof is given in Appendix A. We turn to compare our solution with the network solution denoted "M-P" (Mehta and Patel, 1983, 1986), presumably applied in StatXact (StatXact 5, 2001), and a network variant "V-T" (Valz and Thompson, 1994), reportedly used by SAS (SAS 8, 1999). Both consider only unassigned regions of rectangular shapes (for reasons discussed below). For such cases, M-P add an extra row to complete all column sums to $c_\star$, the maximal one, and find the exact min and max for this table. These, divided by $\prod_j \binom{c_\star}{c_j}$, bound the original min and max. V-T, on the other hand, use the technique we employ in (c) above to bound both min and max, claiming these appear to perform overall comparably to the M-P bounds. Note that both approaches, contrary to ours, cannot be applied to one-way tables.

The bound we obtain on min should be superior to both methods, owing to the well behaved nature of the isotropic real function we globally minimize. Computing the max is a convex maximization problem. This is generally hard since the number of candidate local maxima (polytope edges) grows exponentially in the sample size. Techniques in the spirit of M-P certainly exist (e.g., surveyed by Benson [1995]), but based on our empirical practice, this bound is far less put to use than its counterpart, and thus, as V-T before us, we opt for a bound which is easier to compute.

For unfolding the sample space, the tree and network approaches define opposite views. The tree approach relies on the simplicity of the bounds it computes while allowing itself to solve the same problems many times during the computation. At the other extreme, the network algorithm invests considerably in more complicated topology and computationally intensive data structures (discussed by Valz and Thompson [1994]) to solve each problem only once and then add the solution to all paths arriving at that network node (see Mehta and Patel [1986]). For the M-P bounds, it was claimed by Mehta and Patel (1986) that the network solution is more efficient. But note that in that scenario, each max bound required solving several systems of equations. Utilizing the sharper lower bound we put forth, the computationally cheaper upper bound adapted from V-T, and the convexity of these two allowing further speed-up of a single variable assignment (Section 4.5), the issue merits a revised empirical review, to be presented elsewhere.

7. DISCUSSION

In this work, we present a general framework for efficient computation of exact p-values and similar conditional sums, using a branch-and-bound strategy. We explicitly define the quantities that require swift computation in order to make the approach viable. A successful detailed application is carried for frequency tables, and an outline is provided for novel directions in $r \times c$ contingency tables. By extending the sample space to allow real assignments, we utilize the convexity of many test statistics to obtain tight, easy to compute, lower bounds, as well as further speed-up per each node computation. Assignment order is also explored to improve efficiency. We show that the resulting method significantly extends the practicable range of the exact test for small samples, sparse null hypotheses, and small p-values, all quite common in bioinformatics.

Indeed, several already established bioinformatic tools may benefit from incorporating our method. In Consensus (Hertz and Stormo, 1999), significant patterns are sought in aligned biosequences. The authors define a likelihood ratio statistic and measure the departure of an alignment from a background distribution. Acknowledging that χ^2 approximation is inaccurate in the per column test, the authors invoke a large deviation technique to compute an approximate p-value, which can often be replaced by our exact computation. When evaluating the matches of a profile to a given protein sequence, the Blocks+ curators (Henikoff *et al.*, 1999) score the given profile against many proteins in order to set a significance threshold. EMATRIX (Wu *et al.*, 2000), on the other hand, recursively computes a quantile function, using calculations which in retrospect are similar to ours, to achieve the same goal more rapidly. It would be interesting to apply our method to this problem and compare. The protein profile aligner IMPALA (Schaffer *et al.*, 1999) uses a scaled asymptotic approximation to fit an extreme value distribution to its scores. Recently, Yona and Levitt (2002) used a likelihood ratio statistic to compare column compositions between two profiles in order to optimally align them. The significance of each score was obtained indirectly, and yet the method was shown to surpass IMPALA, especially for remote "twilight zone" homologies. Our method may be applied here to find the exact optimal match. Bulyk *et al.* (2002) have recently demonstrated, through mutation and gene expression measurements, that nucleotides in binding sites can be strongly correlated. This phenomena can be measured in sequence alignments, using the contingency table tests discussed in the previous section. Finally, Carugo and Pongor (2002) define a simple numerical measure of fold similarity between two protein structures in order to allow the scanning of large databases. They compute a histogram of distances between pairs of C^{α} atoms for each protein and compare the two histograms in a hypothesis test that asks whether the two sample come from the same multinomial distribution or not. This question is closely related to the one discussed in the previous chapter. In order to use the chi squared approximation, the authors combine small categories. Our method can allow the computation of the exact p-value of the unaltered sample.

The approach we describe can also be extended in several promising directions. First, in this paper we focused on exact p-value computation. In practice, we often require only a certain amount of accuracy or wish to bound the p-value below a required threshold. It is fairly straightforward to perform further pruning in these cases that allows to compute approximate p-values even faster, while maintaining absolute control over the resulting error bar. Apart from giving us a handle on the trade-off between accuracy and runtime, this approach may also pave the way for handling infinite and continuous sample spaces.

Second, we have focused on a particular traversal scheme over types. This enumeration is easy to define and program, but is not necessarily optimized for the task. Network algorithms invest large efforts to reduce all partial types over the same set of variables into equivalence classes with respect to the yet unassigned

variables. Combined approaches, as well as data-dependent traversal schemes (such as our permuted order of assignment) may further improve computation time to yield a truely competitive evaluation method in a broader spectrum of realistic problems. For example, one can now try to combine the network topology with our bounds for frequency tables, grouping together all partial types at some level with equal remaining counts to assign. In general, one may also try to first traverse partial assignments that correspond more naturally to the structure of the distribution over types. Such a traversal might be able to exploit the fact that this distribution decays exponentially fast as a function of the distance from the null distribution.

Finally, it is our belief that this method can be further exploited in many other statistical computations. Thus, one may apply the real extension approach to Fisher's exact test and other convex statistics. Where analytical solutions cannot be found, one can efficiently use the well established iterative proportional fitting procedure (Darroch and Ratcliff, 1972) or conjugate gradient methods (e.g., Press *et al.*, 1993). Kolmogorov–Smirnov type tests for ordered categorical data also appear to be prone to this approach. Such may also be the case for the wide and computationally intensive areas of bootstrapping, Markov chain Monte Carlo methods, and various permutation tests.

APPENDICES

Appendix A. Proofs

Proof of Lemma 2. Lemma 1 has proven (a) for $\lambda = 0$, since $G^2 \equiv D^{\lambda=0}$. Similarly, we extend our results to all $\lambda > -1$. Again, let $\tau_n \in \mathcal{T}_n^{par}$, and let $\overline{I}$ denote the indices of its yet unassigned categories. Consider the extension of D^λ over the set of all nonnegative real types that sum to n. By differentiating D^λ, we obtain

$$\forall i, j \in \overline{I}: \quad \left[\frac{\partial^2 D^\lambda}{\partial n_i \partial n_j}\right] = \delta_{ij} \frac{2{n_i}^{\lambda-1}}{(nq_i)^\lambda}$$

to conclude, as in Lemma 1, that D^λ is convex for all $\lambda > -1$.

Next, we add a Lagrange multiplier, $J = D^\lambda - \gamma\left(\sum_{i\in\overline{I}} n_i - \overline{n}\right)$, and solve $\nabla J = 0$. For all λ's, we obtain the same solution $\forall i \in \overline{I} : n_i = \frac{q_i}{\overline{q}}\overline{n}$, which by convexity obtains the global real minimum and serves to bound from below the minimum over integer assignments, as in Lemma 1.

To find the maximum of D^λ, we again note that due to the convexity, it must be in one of the extreme points. Let $l \in \overline{I}$ attain the least yet unassigned probability, $q_l = \overline{q}_{min}$. It is easy to check that if $\lambda > -1$, then

$$\forall i \in \overline{I}: \quad \frac{2}{\lambda(\lambda+1)}\,\overline{n}\left[\left(\frac{\overline{n}}{nq_l}\right)^\lambda - 1\right] \geq \frac{2}{\lambda(\lambda+1)}\overline{n}\left[\left(\frac{\overline{n}}{nq_i}\right)^\lambda - 1\right].$$

To see this, note that if $\lambda > 0$, the term $\frac{2}{\lambda(\lambda+1)}$ is positive, and q_l and q_i appear in the denominator, and since $q_l \leq q_i$, we get the desired inequality. When $0 > \lambda > -1$, $\frac{2}{\lambda(\lambda+1)}$ is negative, and q_l and q_i appear in the numerator, and again since $q_l \leq q_i$, we get the desired inequality.

To prove (b), similarly to Section 4.5, which has proven $\lambda = 0$, we set $I = \{1, \ldots, i-1\}$, $\overline{n} = n - \sum_{j=1}^{i-1} n_j$, $\overline{q} = \sum_{j=i+1}^{k} q_j$, and $\overline{q}_{min} = \min_{\{j=i+1,\ldots,k\}} q_j$. The next category assignment variable n_i ranges between $\{0, 1, \ldots, \overline{n}\}$. For $\lambda > 0$ and $-1 < \lambda < 0$, D^λ_{max} and D^λ_{min} have the same form,

$$D^\lambda_{bound}(n_i) = \frac{2}{\lambda(\lambda+1)}\left(\sum_{j=1}^{i} n_j\left(\left(\frac{n_j}{nq_j}\right)^\lambda - 1\right) + (\overline{n} - n_i)\left(\left(\frac{\overline{n}-n_i}{n\overline{q}_\star}\right)^\lambda - 1\right)\right)$$

where $\overline{q}_\star$ denotes $\overline{q}_{min}$ for D^λ_{max}, and $\overline{q}$ for D^λ_{min}. Simple derivation shows this to be convex over $[0, \overline{n}]$ for all these λ's, with a minimum at $n_\star = \frac{q_i}{q_i + \overline{q}_\star}\overline{n}$.

For (c), when $\lambda \leq -1$, the definition of the exact test is more subtle, and likewise its handling. The test is clearly interesting only for samples which are strictly positive ($\forall i : n_i > 0$ ensures $D^\lambda < \infty$). However, samples with zero counts do have a positive probability of occurrence and are by definition always summed into the exact p-value. This property is problematic for our algorithm as nearly all D^λ_{max}'s

turn to infinity, disabling one side of our pruning (albeit, the less effective one). We outline one natural solution to this problem. Split the sample space in two: $\mathcal{T}_n^+$ holding all strictly positive types, and $\mathcal{T}_n^0$ with the rest. Precompute the probability measure of $\mathcal{T}_n^0$ using an inclusion–exclusion summation of all partial types with exactly $1, 2, \ldots, k-1$ zero entries. Next, restrict $[\tau_n]$, and D_{max}^{λ}, D_{min}^{λ} accordingly, to range only over $\mathcal{T}_n^+$. For $|\overline{I}| > \overline{n}$, $[\tau_n]$ is now empty and can be disregarded. For $|\overline{I}| \leq \overline{n}$, one can show that D_{min}^{λ} is obtained as in (a) above. The assignment $n_l = \overline{n} - |\overline{I}| + 1$, and 1's elsewhere can be shown to obtain D_{max}^{λ}. One can also show, as in (b), that D_{max}^{λ}, D_{min}^{λ} are both convex in the next assignment variable n_i over $\mathcal{T}_n^+$. We thus combine the result of our pruning algorithm on $\mathcal{T}_n^+$ with the measure of $\mathcal{T}_n^0$ to obtain the exact p-value. Note that $|\mathcal{T}_n^0| \simeq 2^k$ and thus is typically very small compared to $|\mathcal{T}_n^+|$, which we prune. Moreover, its measure is independent of a particular T_n and can thus be reused for any sample in the same setting. ■

Proof of Lemma 3. For a partial assignment $\tau_{i_0 j_0}$, define the yet unassigned partial column and rows sums $\overline{c}_{j_0} = c_{j_0} - \sum_{i \leq i_0} n_{i j_0}$, and similarly for $\{\overline{r}_i\}_{i=1}^{r}$. Each of these is uniquely determined by the total row and sum margins and the assigned variables. Through these, we express $n_\alpha = \sum_{i \leq i_0} \overline{r}_i$ and $n_\beta = \sum_{i > i_0} \overline{r}_i - \overline{c}_{j_0}$, depicted in Fig. 10.

(a) Denote by $\mathcal{T}$ the reference set of all tables respecting the given row and column margins. Using the chain rule we obtain

$$Q(\tau_{i_0 j_0}) \equiv Q(n_{1c}, n_{2c}, \ldots, n_{i_0 j_0} | \mathcal{T}) = Q(n_{1c}|\mathcal{T}) Q(n_{2c}|\mathcal{T}, n_{1c}) \ldots Q(n_{i_0 j_0}|\mathcal{T}, n_{1c}, n_{2c}, \ldots, n_{i_0 - 1 j_0})$$

where every term on the right can be expressed as a hypergeometric probability. E.g.,

$$Q(n_{i_0 j_0}|\mathcal{T}, n_{1c}, \ldots, n_{i_0 - 1 j_0}) = \frac{\binom{n_{i_0 j_0} + \overline{c}_{j_0}}{n_{i_0 j_0}} \binom{\overline{r}_{i_0} + n_\beta}{\overline{r}_{i_0}}}{\binom{n_{i_0 j_0} + \overline{r}_{i_0} + \overline{c}_{j_0} + n_\beta}{n_{i_0 j_0} + \overline{r}_{i_0}}}.$$

(b) Denote by $\overline{I}$ all unassigned index-pairs for $\tau_{i_0 j_0}$. Simplify G^2 into

$$G^2 = 2 \left(n \log n + \sum_{ij} n_{ij} \log n_{ij} - \sum_i r_i \log r_i - \sum_j c_j \log c_j \right).$$

As a function of the unassigned variables, $\{n_{ij}\}_{ij \in \overline{I}}$, G^2 is clearly convex, as it is a sum of $n_{ij} \log n_{ij}$ terms plus a constant part. Add r Lagrange multipliers for row sums and j_0 column multipliers. Derivation with respect to the unassigned variables yields $\forall ij \in \overline{I}, n_{ij} = A_i B_j$, for some A_i's and B_j's. Consider the following assignment.

$$\forall i > i_0 \quad n_{i j_0} = \frac{\overline{r}_i \overline{c}_{j_0}}{\overline{c}_{j_0} + n_\beta} \qquad \forall ij \in \alpha \quad n_{ij} = \frac{\overline{r}_i c_j}{n_\alpha + n_\beta} \qquad \forall ij \in \beta \quad n_{ij} = \frac{\left(\frac{n_\beta}{\overline{c}_{j_0} + n_\beta}\right) \overline{r}_i c_j}{n_\alpha + n_\beta} \tag{10}$$

It is easy to verify that this assignment respects all marginal sums and can be decomposed as required. Thus, it is one of the local minima. But since our function itself is convex, this must be the global minimum over real assignments.

(c) When one retains only row margins, for example, the function we wish to maximize decomposes into a sum of $i = 1, \ldots, r$ functions, of the same form: maximize $\sum_j n_{ij} \log n_{ij}$ constrained by $\sum_j n_{ij} = \overline{r}_i$. Again, the problem is convex, and its maximum is obtained at an extreme point of the allowed region, where one of the variables is assigned all remaining $\overline{r}_i$ counts. Thus, the maximum respecting only row margins is $\sum_i \overline{r}_i \log \overline{r}_i$, and analogously for the maximum respecting only column margins. These two bound the maximum we seek from above.

(d) The minimal bound we derive above is a sum of $n_{ij} \log n_{ij}$ terms for all variables in (10). When $n_{i_0 j_0}$ increases by some Δn, it affects four quantities in those equations: n_β increases and $\overline{c}_{j_0}$, $\overline{r}_{i_0}$, and n_α decrease by the same amount. Note that the sums $\overline{c}_{j_0} + n_\beta$ and $n_\alpha + n_\beta$ remain fixed, as the relative

changes cancel out. Using these observations, we see that all terms boil down to expressions of the form $k_1(k_2 \pm n_{i_0 j_0}) \log k_1(k_2 \pm n_{i_0 j_0})$, each of which is clearly convex, and thus so is their sum.

At the other end, a change in n_{i0j0} affects only a single variable in each of the two upper bounds we obtained above, $\overline{r}_{i_0}$ in the row constrained and $\overline{c}_{j_0}$ in the column constrained. In both cases, the solution takes the form of $k_1 + (k_2 - n_{i0j0}) \log (k_2 - n_{i0j0})$, for some constants k_1, k_2. Thus, each bound by itself is convex. For each, we can efficiently compute the intersections with the threshold of interest and combine the results to obtain up to two pairs of β, γ points and two distinct "discard" regions flanked by three "descend" regions (see Fig. 3). ■

Appendix B. Notes for the practitioner

Several computational tips which speed-up runtime beyond the didactic code of Fig. 2 follow.

- For reasons of machine accuracy, we did not sum $Q(\tau_n)$ terms (which can be very small) to obtain the exact p-value, but rather collected the logs of these quantities. For this purpose, a useful transformation from $\tilde{x} = \log x$, $\tilde{y} = \log y$ to $\tilde{z} = \log (x + y)$ is

$$\tilde{z} = \tilde{x} + \log (1 + \exp (\tilde{y} - \tilde{x}))$$

  which saves an expensive exponentiation operation, as well as being more accurate since by assuring that $\tilde{x} \geq \tilde{y}$ the log operation is bounded between zero and $\log 2$.
- Since we will be repeatedly evaluating $\log Q(\tau_n)$, $G^2_{max}(\tau_n)$, and $G^2_{min}(\tau_n)$, we have prepared in advance look-up tables for $\{\log q_1, \ldots, \log q_k\}$, $\{\log 1, \ldots, \log n\}$, $\{\log 1!, \ldots, \log n!\}$, $\{\log \overline{q}_1, \ldots, \log \overline{q}_k\}$, and $\{\min_1, \ldots, \min_k\}$. The latter two tables are prepared in correspondence with the assignment order σ and are used as $\overline{q}$ and the index of $\overline{q}_{min}$, respectively.
- Common partial sums in the above equations have been passed down the recursion tree to save re-computing them over and over.
- The log and exp operations in the above equation can, in fact, be replaced by a look-up table of $\log (1 + e^{-z})$ values for $z \geq 0$. Linear interpolation between each two sampled values yields a very good fit (similarly for subtraction when $z \geq 0.02$), with loss of accuracy not far from machine precision. In Section 4.6, this would entail a further three-fold reduction over reported results.
- Finally, on multi-CPU or networked machines, the algorithm is trivially parallelizable. New recursive calls can be distributed between all currently free CPUs, and the (log) probability mass accumulates in one shared variable. Since each call is independent of the others, speed-up gain is expected to be maximal.

ACKNOWLEDGMENTS

The authors wish to thank Ya'acov Ritov, Zvi Gilula, Hanah Margalit, Norman Grover, Gal Chechik, and Ran Gilad-Bachrach for illuminating discussions and valuable comments. The work of GB was done while at the Hebrew University and was partially supported by a grant from the ministry of science, Israel.

REFERENCES

Agresti, A. 1992. A survey of exact inference for contingency tables. *Statist. Sci.* 7(1), 131–177.

Agresti, A. 2001. Exact inference for categorical data: Recent advances and continuing controversies. *Statist. Med.* 20, 2709–2722.

Baglivo, J., Olivier, D., and Pagano, M. 1992. Methods for exact goodness of fit tests. *J. Am. Statist. Assoc.* 87(418), 464–469.

Benson, H.P. 1995. Concave minimization: Theory, applications and algorithms, *in* R. Horst and P. Pardalos, eds., *Handbook of Global Optimization*, 43–148. Kluwer, Amsterdam.

Brownlee, K.A. 1965. *Statistical Theory and Methodology in Science and Engineering*, 2nd ed., Wiley, New York.

Bulyk, M.L., Johnson, P.L., and Church, G.M. 2002. Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors. *Nucl. Acids Res.* 30(5), 1255–1261.

Carugo, O., and Pongor, S. 2002. Protein fold similarity estimated by a probabilistic approach based on C^{α}-C^{α} distance comparison. *J. Mol. Biol.* 315(4), 887–898.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. 1990. *Introduction to Algorithms*, MIT Press, Cambridge, MA.

Darroch, J.N., and Ratcliff, D. 1972. Generalized iterative scaling for log-linear models. *Ann. Math. Stat.* 43(5), 1470–1480.

Ewens, W.J., and Grant, G.R. 2001. *Statistical Methods in Bioinformatics: An Introduction*, Springer, New York.

Gilks, W.R., Richardson, S., and Spiegelhalter, D.J. 1996. *Markov Chain Monte Carlo in Practice*, Chapman and Hall, London.

Harbison, S.P., and Steele, Jr., G.L. 1995. *C A Reference Manual*, 4th ed., Prentice Hall, Englewood Cliffs, NJ.

Henikoff, S., Henikoff, J.G., and Pietrokovski, S. 1999. Blocks+: A non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics* 15(6), 471–479.

Hertz, G.Z., and Stormo, G.D. 1999. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 15(7–8), 563–577.

Hutchinson, T.P. 1979. The validity of the chi-square test when the expected frequencies are small: A list of recent research references. *Comm. Statist.* A8, 327–335.

Lin, J. 1991. Divergence measures based on the Shannon entropy. *IEEE Trans. Inf. Theory* 37(1), 145–151.

Mehta, C.R., and Patel, N.R. 1983. A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables. *J. Am. Statist. Assoc.* 78(382), 427–434.

Mehta, C.R., and Patel, N.R. 1986. Algorithm 643 FEXACT: A fortran subroutine for Fisher's exact test on unordered $r \times c$ contingency tables. *ACM Trans. Mathematical Software* 12(2), 154–161.

Mehta, C.R., and Patel, N.R. 1997. *Exact inference for categorical data*. Manuscript.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. 1993. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, London.

Radlow, R., and Alf, Jr., E.F. 1975. An alternative multinomial assesment of the accuracy of the χ^2 test of goodness of fit. *J. Am. Statist. Assoc.* 70(352), 811–813.

Read, T.R.C., and Cressie, N.A.C. 1988. *Goodness-of-Fit Statistics for Discrete Multivariate Data*, Springer-Verlag, New York.

Rockafellar, R.T. 1970. *Convex Analysis*, Princeton University Press, Princeton, NJ.

Rue, H. 2001. Fast sampling of Gaussian Markov random fields. *J. Royal Statist. Soc. Series B* 63(2), 325–338.

SAS 8. 1999. *STAT User's Guide*, SAS Institute.

Schaffer, A.A., Wolf, Y.I., Ponting, C.P., Koonin, E.V., Aravind, L., and Altschul, S.F. 1999. IMPALA: Matching a protein sequence against a collection of PSI-BLAST-constructed position-specific score matrices. *Bioinformatics* 15(12), 1000–1011.

Senchaudhuri, P., Mehta, C.R., and Patel, N.R. 1995. Estimating exact p values by the method of control variates or Monte Carlo rescue. *J. Am. Statist. Assoc.* 90(430), 640–648.

Siegel, S., and Castellan, N.J. 1988. *Nonparametric Statistics for the Behavioural Sciences*, 2nd ed., McGraw-Hill, New York.

Sokal, R.R., and Rohlf, F.J. 1995. *Biometry*, 3rd ed., Freeman, San Francisco, CA.

StatXact 5. 2001. *User Manual*, Cytel.

Valz, P.D., and Thompson, M.E. 1994. Exact inference for Kendall's S and Spearman's ρ with extension to Fisher's exact test in $r \times c$ contingency tables. *J. Comp. Graphic. Statist.* 3(4), 459–472.

van de Wiel, M. 2001. The split-up algorithm: A fast symbolic method for computing p-values of distribution-free statistics. *Comp. Statist.* 16, 519–538.

Welch, W.J., and Gutierrez, L.G. 1988. Robust permutation tests for matched-pairs designs. *J. Am. Statist. Assoc.* 83(402), 450–455.

Wu, T.D., Nevill-Manning, C.G., and Brutlag, D.L. 2000. Fast probabilistic analysis of sequence function using scoring matrices. *Bioinformatics* 16(3), 233–244.

Yona, G., and Levitt, M. 2002. Within the twilight zone: A sensitive profile-profile comparison tool based on information theory. *J. Mol. Biol.* 315(5), 1257–1275.

Address correspondence to:
Gill Bejerano
Center for Biomolecular Science and Engineering
School of Engineering
1156 High St.
University of California
Santa Cruz, CA 95064

E-mail: jill@soe.ucsc.edu