

Efficient Exact p -Value Computation and Applications to Biosequence Analysis

[Extended Abstract]

Gill Bejerano
School of Computer Science & Engineering
Hebrew University
91904 Jerusalem, Israel
jill@cs.huji.ac.il

ABSTRACT

Like other fields of life sciences, bioinformatics has turned to capture biological phenomena through probabilistic models, and to analyse these models using statistical methodology. A central computational problem in applying useful statistical procedures such as various hypothesis testing procedures is the computation of p -values. In this paper, we devise a branch and bound approach to efficient exact p -value computation, and apply it to a likelihood ratio test in a frequency table setting. By recursively partitioning the sample domain and bounding the statistic we avoid the explicit exhaustive enumeration of all possible outcomes which is currently carried by the standard statistical packages. The convexity of the test statistic is further utilized to confer additional speed-up.

Empirical evaluation demonstrates a reduction in the computational complexity of the algorithm, even in worst case scenarios, significantly extending the practical range for performing the exact test. We also show that speed-up greatly improves the sparser the underlying null hypothesis is; that computation precision actually increases with speed-up; and that computation time is very moderately affected by the magnitude of the computed p -value. These qualities make our algorithm an appealing alternative to the exhaustive test, the χ^2 asymptotic approximation and Monte Carlo samplers in the respective regimes.

The proposed method is readily extendible to other tests and test statistics of interest. We survey several examples of established biosequence analysis methods, where small sample size and sparseness do occur, and to which our computational framework could be applied to improve performance. We briefly demonstrate this with two applications, measuring binding site positional correlations in DNA, and detecting compensatory mutation events in functional RNA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RECOMB'03, April 10–13, 2003, Berlin, Germany.
Copyright 2003 ACM 1-58113-635-8/03/0004 ...\$5.00.

Categories and Subject Descriptors

G.3 [Probability and Statistics]: Statistical Computing;
J.3 [Life and Medical Sciences]: Biology and Genetics

General Terms

Algorithms

Keywords

p -value, exact test, branch and bound, categorical data, real extension, frequency tables.

1. INTRODUCTION

Many statistical procedures routinely used in empirical science require the computation of p -values. A common example is the various hypothesis rejection procedures that compute the p -value of an observed statistic with respect to the null hypothesis (e.g. [9, 23]).

In many scenarios, given an observation of an outcome with a test statistic value t , the exact p -value requires us to sum the probability of all possible outcomes that can assign a test statistic value of t or more. In a finite sample space, one can directly achieve this by scanning all possible outcomes. In most real life problems, however, 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.

In this paper, we develop a branch and bound [5] approach¹ for efficient computation of exact p -values, and apply it to a likelihood ratio test. 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 together. If they all pass the test, or if all fail it, we can handle them without considering each one individually. By careful design, our algorithm performs a systematic examination of all possible groups to ensure exact computation of p -values. The convexity of the statistic play an important role in our analysis.

¹Some fields reserve the term “branch and bound” for optimization problems that seek a single point in the underlying domain. We use it here in a broader context.

We show empirically that this design indeed leads to a decrease in the computational complexity of performing the exact test. We achieve this by reducing the complexity of both the number of cases considered during the computation, as well as the number of mathematical operations performed to compute the reduced sum. These improvements lead to orders of magnitudes faster runtime, and greatly extend the range where an exact test is feasible.

Exact computational inference has received much interest in Statistics and related experimental fields in recent years [1], and branch and bound methods have already had several applications in this context (e.g., [11, 16, 17, 25, 26]). Of these [17] is closest in objective to the current work. That work, however, focuses on conditional inference, and is complicated by imposing a network structure over the problem. As a consequence it cannot be applied to the test we are interested in. Our mathematical and algorithmic analysis is more direct and resembles that of [26]. But their sample space is very different than ours, as is the required analysis.

Small sample, sparse data sets and rare events abound in bioinformatics. Of these the most appropriate for our purposes is the analysis of biosequence data (DNA, RNA, proteins). Searches for patterns in aligned sequence data, matches of probabilistic profiles within a given sequence, optimal alignments of sequences and profiles, all pose statistical tests of categorical data. In many cases, such as the analysis of known binding sites, little data may be available. In pattern matching data may be available but inherently sparse, or one may want to search through many combinations, searching for far more rare events than in the single test scenario. Such is the case when examining biologically correlated positions in aligned sequences and discovering that some combinations are mostly avoided. As asymptotic approximations are known to be inexact in these regimes, their use can affect a correct analysis of “twilight zone” instances, where significance may be missed, and random matches may be wrongly pursued. After developing the method, we point out several established bioinformatics methods where our approach may be of use, and demonstrate two applications.

2. A LIKELIHOOD RATIO TEST

Let X be a discrete random variable with a finite set of possible values, or categories $\{1, 2, \dots, k\}$. Let Q be a multinomial distribution over this set, $Q = (q_1, q_2, \dots, q_k)$. Through statistical inference we would like to decide between two hypotheses: either X is distributed according to Q , or it is not. We call the first our null hypothesis, H_0 , and the other its alternative, H_1 . We will base our decision on a set of n independent observations of X . A common relevant example is the composition of a column of multiply aligned biosequences.

Let $S_n = \{x_1, x_2, \dots, x_n\}$ denote the sample. Let T_n denote its empirical *type* which counts how many times each possible value appeared in the sample, $T_n = (n_1, n_2, \dots, n_k)$ where each $n_i = |\{j | x_j = i\}|$. Denote its empirical probability distribution by $P_n = (p_1, p_2, \dots, p_k) = (\frac{n_1}{n}, \frac{n_2}{n}, \dots, \frac{n_k}{n})$.

Our aim is to quantify the extent to which the observed type deviates from the null hypothesis distribution. To this

purpose we define the widely used *likelihood ratio statistic*

$$G^2 = -2 \log \lambda = -2 \log \frac{P(S_n | X \sim Q)}{P(S_n | X \sim P_n)} = 2 \sum_{i=1}^k n_i \log \frac{n_i}{n q_i} \quad (1)$$

where λ is the ratio between the likelihood of the data given H_0 , and its likelihood given the most likely distribution in H_1 (see, e.g., [9] for more details). Our statistic is thus a real valued function from the *sample space*, the collection of all possible empirical types of size n :²

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

We denote by $d_n = G^2(T_n)$, the value the statistic attains for a given sample. We can now define a hypothesis test for the likelihood ratio statistic of a given sample:

1. Compute $d_n = G^2(T_n)$;
2. Reject H_0 iff $d_n \geq$ a predetermined threshold.

Two types of errors are possible in this test. Type *I* error concerns rejecting H_0 when it is true, and type *II* error concerns not rejecting H_0 when it is false. The p -value of a test is defined as the minimal bound on a type *I* error (or confidence level) for which we would reject H_0 given S_n . For our decision criterion it boils down to

$$p\text{-value} = \Pr(G^2(T'_n) \geq d_n | X \sim Q) \quad (2)$$

Thus, the p -value is the probability to draw under H_0 a sample of size n , for which the chosen discrepancy measure is at least as large as that of our observed sample. This quantity, however, is not easily obtained.

3. MEASURES OF P-VALUE

3.1 Exact Test

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

$$Q(T_n) = n! \prod_{i=1}^k \frac{q_i^{n_i}}{n_i!} \quad (3)$$

The direct approach to compute (2), taken by the standard software packages [20, 24], explicitly sums

$$p\text{-value} = \sum_{\substack{T'_n \in \mathcal{T}_n \text{ s.t.} \\ G^2(T'_n) \geq d_n}} Q(T'_n) \quad (4)$$

by examining all possible types of size n . This approach is only practicable 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.³

A theoretically efficient exact alternative computes the characteristic function of the chosen statistic and then inverts it using the fast Fourier Transform (e.g., [2]). However, its use of trigonometric functions and complex arithmetics deems it too inaccurate for practical use.

²We did not need the space of all possible samples because a type is a jointly sufficient statistic of an i.i.d. sample.

³Note, however, that $|\mathcal{T}_n| \leq (n+1)^k$, whereas the alternative space we could sum over, of all possible samples, is of size k^n , and is typically much larger.

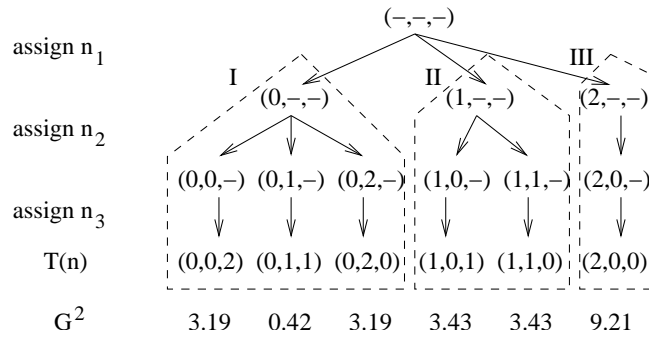


Figure 1: Exact test recursion tree. We recursively develop all possible types for $k=3$ and $n=2$, by assigning all allowed values to each category in turn. Below the tree we write each type’s G^2 statistic for $Q = (.1, .45, .45)$.

3.2 Asymptotic Approximation

Under broad regularity conditions G^2 can be shown to have a well known and easily computed asymptotic form. Namely, with $n \rightarrow \infty$, G^2 converges to a χ^2 distribution with $k-1$ degrees of freedom (see [9]), giving rise to the widely used chi-squared approximation

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

However convergence rate and direction for G^2 vary with the choice of parameters in H_0 . Thus the approximation is problematic for small sample size and minute p -values. Bigger samples that fall outside the regularity conditions are also problematic. Such is *sparseness* in the expected type, commonly defined as a test where $nq_i \leq 5$ for at least one index i . Unfortunately these regions are quite common in bioinformatics. Correcting factors and improved approximations of the distribution of G^2 offer limited added guaranty. Until recently practitioners have mainly used heuristics to circumvent these obstacles, such as merging or ignoring sparse categories and then applying the asymptotic approximation (see, e.g., [23]), suffering potentially harmful accuracy tolls.

3.3 Simulation

With the advent of computing power, Monte Carlo methods using computer simulations have become widely practiced. A simple simulation approach to estimate (2) is to draw R (pseudo-random) i.i.d. samples, $\{S_n^{(1)}, \dots, S_n^{(R)}\}$, according to distribution Q , and approximate

$$p\text{-value} \simeq \frac{|\{r \mid G^2(T_n^{(r)}) \geq d_n\}|}{R}$$

Theoretical bounds on the variance of the estimate, such as Hoeffding’s inequality [7], can quantify the amount of required observations. However, smaller p -values and more accurate estimates require a great number of samples, of the order of $p\text{-value}^{-1}$.

4. EFFICIENT EXACT P-VALUE COMPUTATION

4.1 Motivation

Consider a simple example for $k=3$ categories and $n=2$ observations. Assume we expect $Q = (.1, .45, .45)$, but sample $T_n = (1, 0, 1)$. What is the p -value of such a finding?

One way to explicitly enumerate all possible types in order to perform the exact test (4) 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 , etc. At the leafs of the recursion tree we find all 6 possible assignments. We can calculate G^2 for each and accumulate the mass of those who are at least as big as $d_n = G^2(T_n) \simeq 3.43$.

Note, however, that if we had tight upper and lower bounds on the values G^2 obtains in a given sub-tree, we could have ended our computation after assigning only n_1 : The maximal G^2 value in sub-tree I , falls below d_n and thus this whole sub-tree can be discarded. While the minimal G^2 value in sub-trees 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 3 nodes, and conclude with the exact answer. We turn to formalize and extend these intuitions.

4.2 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, \dots, 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

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

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

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

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

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

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

We define the probability of τ_n , under the null hypothesis,

Call format:
 $p\text{-value} = \text{descend}(1, (-, \dots, -))$

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

```

{
  p-val = 0
  for  $n_i = 0$  to  $n - \sum_{j=1}^{i-1} n_j$ 
    {
       $\tau_n = (n_1, \dots, n_i, -, \dots, -)$ 
      if  $G_{min}^2(\tau_n) \geq d_n$ 
        p-val = p-val +  $Q(\tau_n)$ 
      else if  $G_{max}^2(\tau_n) \geq d_n$ 
        p-val = p-val + descend( $i + 1, \tau_n$ )
    }
  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}_σ from the empty assignment root node. For each node, or partial assignment τ_n :

Pruning Criterion 1. If $G_{min}^2(\tau_n) \geq d_n$, 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 $G_{max}^2(\tau_n) < d_n$, 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.

Figure 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.

as the sum of the probabilities of all types in $[\tau_n]$,

$$Q(\tau_n) = \sum_{T_n \in [\tau_n]} Q(T_n) = n! \frac{\bar{q}^{\bar{n}}}{\bar{n}!} \prod_{i \in I} \frac{q_i^{n_i}}{n_i!} \quad (5)$$

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}_σ , is a tree where the root node contains the empty assignment $(-, \dots, -)$. 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. 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 sub-tree 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 sub-domains, by defining

$$G_{max}^2(\tau_n) = \max_{T_n \in [\tau_n]} G^2(T_n), \quad G_{min}^2(\tau_n) = \min_{T_n \in [\tau_n]} G^2(T_n)$$

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

$$G_{max}^2(\tau_n) = 2 \left(\sum_{i \in I} n_i \log \frac{n_i}{n q_i} + \bar{n} \log \frac{\bar{n}}{n \bar{q}_{min}} \right) \quad (6)$$

$$G_{min}^2(\tau_n) \geq 2 \left(\sum_{i \in I} n_i \log \frac{n_i}{n q_i} + \bar{n} \log \frac{\bar{n}}{n \bar{q}} \right) \quad (7)$$

PROOF. Let $\tau_n \in \mathcal{T}_n^{par}$, and let \bar{I} denote the indices of its yet unassigned categories. Consider the extension of G^2 over the set of all non-negative *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,

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

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 (see [19], p. 27).

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

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

By solving $\nabla J = 0$ we obtain the solution $\forall i \in \bar{I}: n_i = \frac{q_i}{\bar{q}} \bar{n}$. Since G^2 is strictly convex, this interior point must be a global minimum (see [19], p. 242). In general this will not yield a valid integer assignment but it does bound G_{min}^2 from below, obtaining (7).

Since G^2 is convex, it achieves its maximum value in extreme points of the allowable region (see [19], 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 $l \in \bar{I}$ attain the least yet unassigned probability, $q_l = \bar{q}_{min}$. Clearly,

$$\forall i \in \bar{I}: \log \frac{\bar{n}}{n q_l} \geq \log \frac{\bar{n}}{n q_i}$$

Thus, assigning all \bar{n} remaining counts to n_l , is the maximal value of G^2 in $[\tau_n]$, yielding (6). \square

To simplify notations we will next use the right side of (7) as the value of G_{min}^2 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 Algorithm

We will now utilize the domain partitioning as the “branch” step. The easily computable bounds on the statistic in a given partition, and probability measure of the partition comprise the “bound” step. The resulting algorithm is shown in Fig. 2.

Note 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 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 have yet to be resolved. This

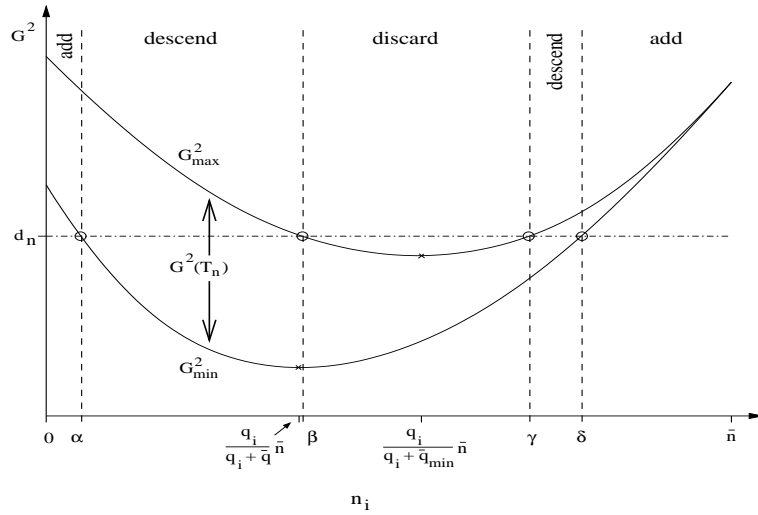


Figure 3: Faster recursion step at the node level. We plot the values of G_{max}^2 and G_{min}^2 vs. all possible real assignments to the next category n_i . A threshold d_n can intersect each of the two convex curves at most twice. The four intersection values, denoted $\alpha - \delta$ define five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

situation is common when performing multiple combination tests. The above observation is also an easy starting point to critical value computation and generation of a look-up table in a single traversal.

4.5 Faster 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_{max}^2 and G_{min}^2 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., $G_{min}^2(\tau_n) < d_n \leq G_{max}^2(\tau_n)$). Denote, with a slight abuse of notation, $\bar{n} = n - \sum_{j=1}^{i-1} n_j$, $\bar{q} = \sum_{j=i+1}^k q_j$ and $\bar{q}_{min} = \min_{\{j=i+1, \dots, k\}} q_j$. We need to assign to the next category n_i all possible values $\{0, 1, \dots, \bar{n}\}$, and examine G_{max}^2, G_{min}^2 for each. Since both bounds (6), (7) have the same form, as a function of n_i , we can write compactly:

$$G_{bound}^2(n_i) =$$

$$2 \left(\sum_{j=1}^{i-1} n_j \log \frac{n_j}{n q_j} + n_i \log \frac{n_i}{n q_i} + (\bar{n} - n_i) \log \frac{\bar{n} - n_i}{n \bar{q}_*} \right)$$

where $G_{bound}^2 = G_{max}^2$ for $\bar{q}_* = \bar{q}_{min}$ and $G_{bound}^2 = G_{min}^2$ for $\bar{q}_* = \bar{q}$. It is easy to verify that G_{bound}^2 is strictly convex in n_i when $n_i \in [0, \bar{n}]$, and obtains its minimum at $n_* = \frac{q_i - \bar{q}_*}{q_i + \bar{q}_*} \bar{n}$ (which is in general fractional). Since by definition $\bar{q}_{min} \leq \bar{q}$ the “swoosh”-like shape of the two bounds as a function of n_i is as shown in Fig. 3.⁴ Clearly, any threshold d_n can intersect either curve at most twice. The four (or less) intersection

⁴Whereas the vertical ordering of points $G_{max}^2(n_i = \bar{n}) = G_{min}^2(n_i = \bar{n})$, $G_{max}^2(n_i = 0)$ and $G_{min}^2(n_i = 0)$ is inverse to that of q_i , \bar{q}_{min} and \bar{q} , respectively.

values, denoted $\alpha - \delta$ in Fig. 3, define five (or less) groups of n_i values. All values in each group are equally treated. Based on our analysis of G_{bound}^2 we can now perform four *binary searches* (see, e.g., [5]), to elucidate $\lfloor \alpha \rfloor$, $\lceil \beta \rceil$, $\lfloor \gamma \rfloor$ and $\lceil \delta \rceil$, by searching for d_n with G_{max}^2 and G_{min}^2 over their respective $\{0, \dots, \lfloor n_* \rfloor\}$ and $\{\lfloor n_* \rfloor + 1, \dots, \bar{n}\}$ (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 . More precisely, we have reduced the examination work performed at each node from $\Theta(\bar{n})$ in Fig. 2 to $\Theta(\log \bar{n})$.

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.

Thus, while performing exactly the same recursive calls as the algorithm of Fig. 2, this variant reduces the amount of time spent in each invocation of the procedure.

4.6 Computational Complexity

Five computational procedures were implemented in C, and run on a Pentium III 733MHz Linux machine, employing further practical speed-ups (see Appendix):

- **Direct** - exact computation by recursive enumeration of all types, as described in Sec. 3.1. Equivalent procedures are performed by SAS [20] and StaXact [24].
- **Pruned** - exact computation by recursive enumeration with the two pruning criteria of Sec. 4.4.
- **Convex** - same as Pruned but exploits the convexity of the bounds on G^2 , as in Sec. 4.5.
- χ^2 - the chi-squared approximation discussed in Sec. 3.2 (computed as in [18]).

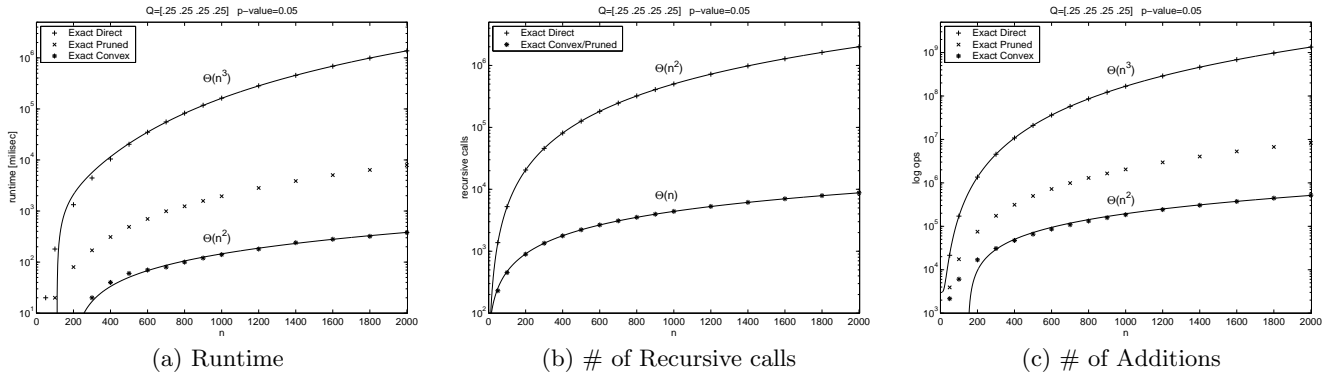


Figure 4: Performance evaluation of the exact algorithms. All three graphs plot the cost of p -value computation for $Q = (.25, .25, .25, .25)$ with different choices of n . The x -axes denote the number of samples, n . The y -axis denotes computation cost, using three different performance measures. 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 show in Fig. 5 to be our worst case in terms of performance.

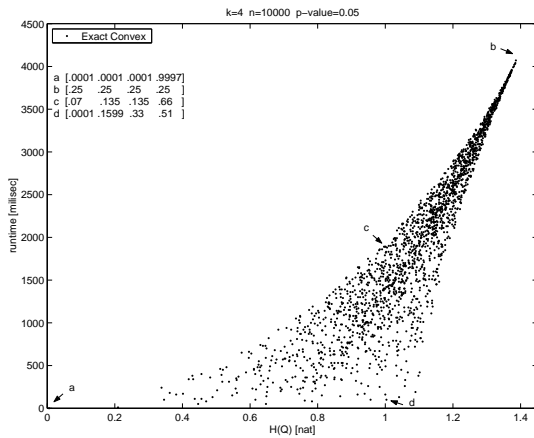


Figure 5: The correlation between the entropy of Q (x -axis) and the runtime of computing p -values (y -axis). The points correspond to 2000 different Q 's picked uniformly from the space of all four categories distributions. Four points are labeled: (a),(b) at the two extremes of distribution entropy 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, point (b), computed in 4 seconds using Convex, would require about 55 hours using Direct.

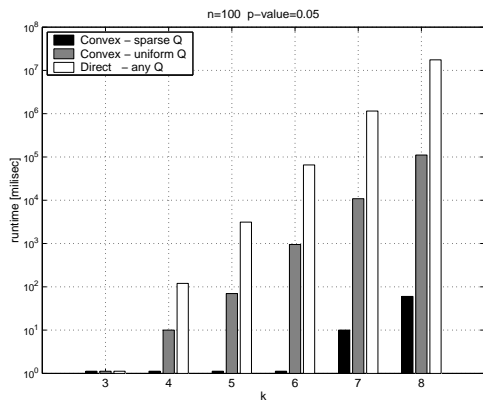


Figure 6: Effect of the number of categories (x -axis) on the runtime of the algorithm (y -axis). The figure compares the increase in runtime for Direct and Convex when computing a p -value of 0.05 with $n = 100$. Convex computation time ranges between that for sparse Q 's (here $n \cdot q_i = 1$ for all $i \neq k$) and for a uniform Q . For Direct all Q 's require about the same computation time.

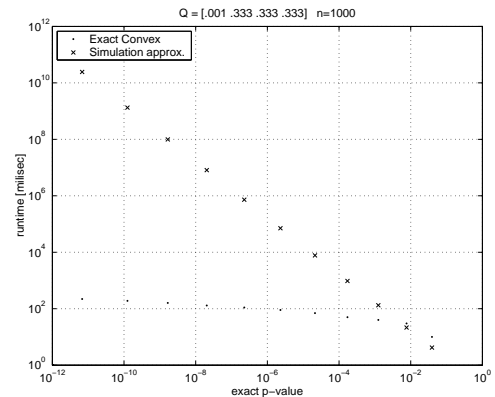


Figure 7: Computation of small p -values. For $Q = (0.001, 0.333, 0.333, 0.333)$ we plot the exact p -value of the rounded types ($i, \frac{1000-i}{3}, \frac{1000-i}{3}, \frac{1000-i}{3}$) for $i = 5, \dots, 15$, (x -axis, right to left). For each we plot the actual runtime of Convex, and an extrapolation 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).

- **Simulation** - the Monte-Carlo sampler approximation from Sec. 3.3.

An extensive evaluation of the different algorithms has been carried out, but due to space limitations will await the full version. We summarize our major findings:

For fixed k and growing n , $|\mathcal{T}_n| \simeq \frac{n^{k-1}}{(k-1)!}$. In Fig. 4 we measure the performance of the first three exact algorithms in such a scenario. Using polynomial fitting we observe that the complexity decreases by one degree in all measured categories: runtime, recursive calls, and mathematical operations. As we next show in Fig. 5 the uniform scenario we examine here is the one where our speed-up is least effective. And yet, even there it lowers the polynomial complexity rather than just improve the constants. Other tests we have performed indicate that the degree gap in fact grows in our favor for larger k .

The summation of whole sub-trees we perform via (5) is also much more accurate in terms of machine precision than the exhaustive enumeration **Direct** performs, that sums all types within every sub-tree. Thus, our algorithm is both more efficient and more precise than the one in use today. And in general the more efficiently it runs, the more complete sub-trees it sums, and the more precise its end result.

For a non-uniform null hypothesis Q , the preferred traversal order σ in **Convex** appears to always be from smallest q_i to largest. We have observed this behaviour in every one of the many diverse scenarios examined, and will shortly offer a plausible explanation for it. All subsequent reported results for **Convex** adopt this traversal scheme.

As Fig. 5 shows, speed-up was found to increase the lower the entropy [6] of the null distribution is, and among equals it further increases the sparser Q is. This makes our algorithm a preferred alternative to the asymptotic χ^2 which is known to be inaccurate in this regime.

Tying the last two results together, note that even for relatively small resulting p -values, most types are usually summed in (4). The value of G^2 for all these exceeds d_n . By expanding lower q_i values first we add bigger $n_i \log \frac{n_i}{n q_i}$ terms first into the accumulating G^2 value, and thus cross the d_n threshold sooner, to invoke pruning criterion 1 earlier on. This speedup becomes all the more evident the more small q_i values there are in sparse null distributions.

Despite these desirable properties, as the number of categories k in a test increases, so does the sample space, and with it, inevitably, our runtime. In Fig. 6 we demonstrate this effect by comparing for a growing k our best runtime, for a sparse $Q = (\frac{1}{n}, \dots, \frac{1}{n}, \frac{n-k+1}{n})$; our worst, when Q is uniform; and the runtime of **Direct**, which is about the same for all null distributions.

But while the tests performed above all focus on the ubiquitous 0.05 threshold, in many cases where data abounds, especially in molecular biology, we are interested in running many different test combinations, searching for the most surprising of these. However, the more tests one performs the more likely it is to find spurious hits with seemingly low p -values. The threshold for interest is thus significantly lowered in these scenarios, and searches for p -values of the order of 10^{-6} or even 10^{-10} are not uncommon. A sampler such as the Monte Carlo **Simulation** will need to sample inversely proportional to the p -value it tries to measure. Here we find another useful property of our exact algorithm. As exemplified in Fig. 7 computation time for **Convex** is very moderately

affected by the size of the resulting p -value, allowing it to compute exact values in regimes where a sampler run for a comparable amount of time would yield unacceptably high estimation variance.

5. APPLICATIONS

5.1 Brief Survey

We highlight several established tools in biosequence analysis where our method may improve performance.

In **Consensus** [15] significant patterns are sought in aligned sequences. They define a likelihood ratio statistic and measure the departure of an alignment from a background distribution. Acknowledging that χ^2 approximation is inaccurate in their regime they resort to large deviation approximation and intensive numerical simulation to achieve an approximate p -value. As columns are treated independently it appears that our method could be applied per-column, to later combine the resulting p -values, e.g., as in [3].

When evaluating the matches of a profile to a given sequence, the **Blocks+** curators [13] score the given profile against many proteins in order to set a significance threshold for it. While **EMATRIX** [27] recursively computes a quantile function, using calculations which in retrospect are similar to ours, to achieve the same goal much more rapidly. It would be interesting to apply our method to this problem and compare.

The profile aligner **IMPALA** [21] uses a scaled asymptotic approximation to fit an extreme value distribution to their scores. Recently, in [28] a likelihood ratio statistic was used 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 can be applied here to find the exact optimal match.

Finally, in [4] it was 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 a variant of our test, as we next demonstrate.

5.2 Binding Site Correlations

The RNA polymerase responsible for transcription initiation in *Escherichia coli* has two functional components: the core enzyme and the sigma factor. *E. coli* has several different σ -factors, each conferring to the polymerase specificity to different promoter sequences [12].

We have chosen a set of nitrogen regulated gene promoters, controlled by σ^{54} , from the PromEC database [14]. The set was aligned according to known promoter signatures to create an alignment block. Every pair of columns in the block was subjected to a test of independence: The distribution of bases in each column was computed separately, $\{p_A, p_C, p_G, p_T\}$ for column i , and $\{q_A, q_C, q_G, q_T\}$ for column j . The null hypothesis claimed that the expected distribution of pairs of bases is that of two independent columns, i.e., f_{AC} the expected frequency of observing A in column i and C in column j is $p_A q_C$. Our algorithm was used to assign an exact p -value to the actual counts in each pair of columns. These were then sorted in increasing order. In principle, now must follow a correction term for performing multiple tests (such as Bonferroni’s). For our purposes it

rank	exact p -value	χ^2 approx.	pos.	observed combinations	missing combinations
1	0.016	0.002	(1,2)	7AA, 2TT	AT, TA
2	0.051	0.006	(5,24)	4TA, 3CT, 1CC, 1GC	TT, CA
3	0.153	0.017	(1,22)	6AT, 2TA, 1AA	TT

Table 1: *E. coli* promoter positional independence test. The σ^{54} results are derived from an alignment of 9 sequences. The consensus of the examined block of 24 columns is N(5)-TGGCAC-N(5)-TTGC-N(4). Positions are given with respect to the pattern, starting from 1. In the observed combinations column, the entry “4TA” denotes 4 sequences with T in pos. 5 and A in pos. 24. Missing combinations refers to highly expected combinations according to the independence assumption. The χ^2 degrees of freedom were corrected to account for parameter estimation from the data (see [23]).

suffices to examine the top three scoring p -values, reported in Table 1.

The top matches occur outside the consensus sequence. These flanking regions, while much less conserved, are known to play an important role in promoter recognition, and their analysis poses a challenge, partly resolved by such tests. Also note the inaccuracy of the χ^2 approximation, which in most analyses would have wrongly highlighted the third ranking combination, as well as others further down the list.

5.3 Compensatory Mutations

Homologous biosequences are known to sometimes undergo compensatory mutation events. Such is often the case in functional RNA, where matching base pairs form hydrogen bonds that stabilize the RNA tertiary structure [8]. When a base undergoes mutation that disrupts the complementarity of a paired base, a mutation in the second base to regain complementarity is usually favoured energetically.

Searching to find such events we examined the family of Vault RNAs. These are part of the vault ribonucleoprotein complex, encoding a complex of proteins and RNAs suspected to be involved in drug resistance [22]. A 146 base-long gapped alignment of 16 family members was taken from the Rfam database [10, ref. RF00006]. Using the independence test described above, the 6 top ranking column pairs were found to be all combinations of columns (4,5,142,143). The observed combinations in these were 9CAUG, 7UGCA. The complementarity of cols. (4,143) and (5,142) in both combinations, and lack of any other combination suggest a compensatory event of two interacting base-pairs. This information can be used to guide or validate RNA structure prediction as well as phylogenetic reconstruction algorithms (see [8]).

To further demonstrate the inaccuracy of the χ^2 approximation in such tests we show in Fig. 8 the relative error of this approximation compared to the exact values, for all vault RNA tests resulting in p -values in the critical range – where wrong values lead to false rejection or acceptance.

6. DISCUSSION

In this paper we show how to perform a branch and bound efficient computation of exact p -values for a goodness of fit likelihood ratio test, utilizing the convexity of our test statistic. We empirically showed that this approach reduces the computational complexity of the test, as well as improve precision over the exhaustive approach in use today. Other useful properties of our approach include increased speed-up gain for sparse null hypotheses, very moderate increase in run time the smaller the resulting p -value is, and the ability to compute p -values for several thresholds simultaneously.

Our method thus significantly extends the practicable range of the exact test for small samples, sparse nulls, and small p -values, all quite common in bioinformatics.

It is clear that the same methodology can be adopted to other tests and test statistics of interest. For example, the widely used Pearson X^2 statistic is also convex, and can undergo the very same treatment. Such appears to be the case for the established bioinformatics tools surveyed in Sec. 5. Of these perhaps the most promising are the methods which combine several p -values through dynamic programming or related methods to arrive at an optimal configuration. We have also demonstrated how such tests can examine biological hypotheses, such as the spread of binding site correlations, and guide or help evaluate algorithms for RNA structure prediction and phylogenetic inference.

A promising direction to further improve runtime involves the machine-precision accuracy of our procedure. In practice, we often only require a certain amount of accuracy or just 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. Such approximations can then provide tight pre-defined control on the tradeoff between accuracy and runtime. They may also allow us to extend the general method to infinite and continuous sample spaces.

Finally, in this paper we focused on a particular depth-first traversal scheme over types. This enumeration is easy to define, but is not necessarily optimized for the task. A potential way of performing more efficient computation is by a breadth-first traversal of partial assignments that corresponds more naturally to the structure of the distribution over types, as the later is known to decay exponentially fast as a function of the distance from the null hypothesis (see [6]). This direction may lead to a further reduction in time complexity of the exact computation.

7. ACKNOWLEDGMENTS

The author is indebted to Nir Friedman and Naftali Tishby for valuable discussions, and to the referees for constructive comments. M.N. and Y.F. are acknowledged for upsetting the author into initial interest in the subject matter. This research was supported by a grant from the Ministry of Science, Israel.

8. REFERENCES

- [1] A. Agresti. Exact inference for categorical data: recent advances and continuing controversies. *Statist. Med.*, 20:2709–2722, 2001.

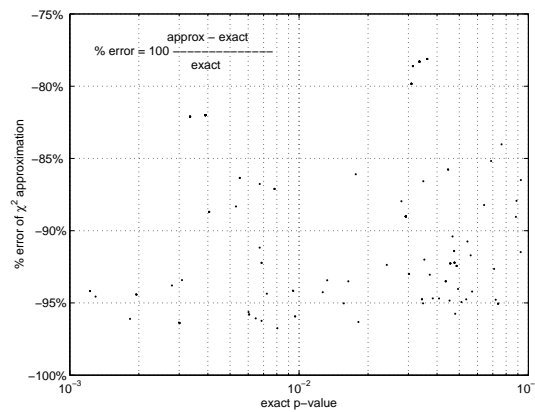


Figure 8: Comparison of the exact and approximate p -values of the vault RNA data. For all column pair combination tests with resulting exact p -values in the critical range between 0.001 and 0.1, we plot the exact p -value (x -axis) against the relative error of a χ^2 approximate value for the same test (y -axis). The corrected approximation (as in Table 1) is shown to under-estimate the p -value considerably.

- [2] J. Baglivo, D. Olivier, and M. Pagano. Methods for exact goodness of fit tests. *J. American Statistical Assoc.*, 87(418):464–469, 1992.
- [3] T. L. Bailey and M. Gribskov. Combining evidence using p -values: application to sequence homology searches. *Bioinformatics*, 14(1):48–54, 1998.
- [4] M. L. Bulyk, P. L. Johnson, and G. M. Church. Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors. *Nucleic Acids Res.*, 30(5):1255–1261, 2002.
- [5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT, 1990.
- [6] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley, 1991.
- [7] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*. Springer, 1996.
- [8] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge, 1998.
- [9] W. J. Ewens and G. R. Grant. *Statistical methods in bioinformatics: an introduction*. Springer, 2001.
- [10] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy. Rfam: an RNA family database. *Nucleic Acids Res.*, 31(1):439–441, 2003.
- [11] D. J. Hand. Branch and bound in statistical data analysis. *Statistician*, 30(1):1–13, 1981.
- [12] J. D. Helmann. Promoters, sigma factors, and variant RNA polymerases. In S. Baumberg, editor, *Prokaryotic Gene Expression*, chapter 3. Oxford, 1999.
- [13] S. Henikoff, J. G. Henikoff, and S. Pietrokovski. Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15(6):471–479, 1999.
- [14] R. Hershberg, G. Bejerano, A. Santos-Zavaleta, and H. Margalit. PromEC: An updated database of Escherichia coli mRNA promoters with experimentally identified transcriptional start sites. *Nucleic Acids Res.*, 29(1):277, 2001.
- [15] G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7-8):563–577, 1999.
- [16] C. R. Mehta and N. R. Patel. A network algorithm for performing Fisher’s exact test in $r \times c$ contingency tables. *J. American Statistical Assoc.*, 78(382):427–434, 1983.
- [17] C. R. Mehta, N. R. Patel, and L. J. Wei. Constructing exact significance tests with restricted randomization rules. *Biometrika*, 75(2):295–302, 1988.
- [18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C: the art of scientific computing*. Cambridge, 2nd edition, 1993.
- [19] R. T. Rockafellar. *Convex analysis*. Princeton, 1970.
- [20] SAS 8. *STAT user’s guide*. SAS Institute, 1999.
- [21] A. A. Schaffer, Y. I. Wolf, C. P. Ponting, E. V. Koonin, L. Aravind, and S. F. Altschul. IMPALA: matching a protein sequence against a collection of PSI-BLAST-constructed position-specific score matrices. *Bioinformatics*, 15(12):1000–1011, 1999.
- [22] G. L. Scheffer, A. C. Pijnenborg, E. F. Smit, M. Muller, D. S. Postma, W. Timens, P. van der Valk, E. G. de Vries, and R. J. Scheper. Multidrug resistance related molecules in human and murine lung. *J. Clin. Pathol.*, 55(5):332–339, 2002.
- [23] R. R. Sokal and F. J. Rohlf. *Biometry*. Freeman, 3rd edition, 1995.
- [24] StatXact 5. *User manual*. Cytel, 2001.
- [25] M. van de Wiel. The split-up algorithm: a fast symbolic method for computing p -values of distribution-free statistics. *Computational Statistics*, 16:519–538, 2001.
- [26] W. J. Welch and L. G. Gutierrez. Robust permutation tests for matched-pairs designs. *J. American Statistical Assoc.*, 83(402):450–455, 1988.
- [27] T. D. Wu, C. G. Nevill-Manning, and D. L. Brutlag. Fast probabilistic analysis of sequence function using scoring matrices. *Bioinformatics*, 16(3):233–244, 2000.
- [28] G. Yona and M. Levitt. Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. *J. Mol. Biol.*, 315(5):1257–75, 2002.

APPENDIX

A. NOTES FOR THE PRACTITIONER

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

- 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 $\log Q(\tau_n)$, $G_{max}^2(\tau_n)$ and $G_{min}^2(\tau_n)$ we have prepared in advance look-up tables for $\{\log q_1, \dots, \log q_k\}$, $\{\log 1, \dots, \log n\}$, $\{\log 1!, \dots, \log n!\}$, $\{\log \bar{q}_1, \dots, \log \bar{q}_k\}$ and $\{\min_1, \dots, \min_k\}$. The latter two tables are prepared in correspondence with the assignment order σ , and are used as \bar{q} and the index of \bar{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 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 Sec. 4.6 this would entail a further 3-fold reduction in runtime over reported results.
- Finally, note that in a multi-CPU or network environment, the algorithm is trivially parallelizable.