Exact Calculation of Distributions on Integers, with Application to Sequence Alignment

Lee A. Newberg[1,2] Charles E. Lawrence[3]

[1]Wadsworth Center, New York State Department of Health

[2]Department of Computer Science, Rensselaer Polytechnic Institute

[3]Division of Applied Mathematics, Brown University

Theory Seminar, Department of Computer Science, Rensselaer Polytechnic Institute, April 29, 2009.

Abstract

- **Background:** Often dynamic programming algorithms seek an optimal integer score, but entire distribution of scores can be of interest.
- **Results:** Three ways to compute distribution of scores. Applied to pairwise alignment of DNA sequences.
- **Conclusions:** Serial algorithm has no increased memory requirement. Highly parallelizable. Credibility $\neq$ statistical significance.

Pairwise Sequence Alignment

Inputs

- x, y: Two strings of letters. Alphabet = $\{A, C, G, T\}$.
- Transform x into y. (SSEARCH) Scores are
  - Match: $+5$
  - Substitution: -4
  - Insertion/Deletion start: -16. Indel extension: -4.

Outputs

- Score of optimal alignment. Viterbi67, NW70, SW81.
- Scores of all alignments. NL09.

Direct Approach

Unaltered Algorithm (Simplified)

Algorithm's typical step looks something like:

$$S(i,j) = \max \left\{ \begin{array}{c} S(i-1,j-1) + M(x_i, y_j), \\ S(i-1,j) + D(x_i), \\ S(i,j-1) + I(y_j) \end{array} \right\}$$

Want $S(m,n)$, where m and n are input strings' lengths.

Direct Approach

Recap: Unaltered Algorithm

$$S(i,j) = \max \left\{ \begin{array}{c} S(i-1,j-1) + M(x_i, y_j), \\ S(i-1,j) + D(x_i), \\ S(i,j-1) + I(y_j) \end{array} \right\}$$

Score Distribution via the Direct Approach

Number of ways to get score s. Typical step:

$$\begin{aligned} Z(i,j,s) \quad = \quad & Z(i-1,j-1,s-M(x_i,y_j)) + \\ & Z(i-1,j,s-D(x_i)) + \\ & Z(i,j-1,s-I(y_j)) \end{aligned}$$

Want $Z(m,n,s)$ for all possible scores s.
Requires increased runtime and memory.

Polynomial Approach

Recap: Score Distribution via the Direct Approach

$Z(m,n,s)$ is number of ways to get score s. Use:

$$\begin{aligned} Z(i,j,s) &= Z(i-1,j-1,s-M(x_i,y_j)) + \\ &\quad Z(i-1,j,s-D(x_i)) + Z(i,j-1,s-I(y_j)) \end{aligned}$$

Score Distribution via the Polynomial Approach

$P(i,j)$ is a "polynomial" in indeterminant ω that "packs" the $Z(i,j,s)$ values. Define $P(i,j) = \sum_s Z(i,j,s)\omega^s$. Typical step:

$$\begin{aligned} P(i,j) &= P(i-1,j-1)\omega^{M(x_i,y_j)} + \\ &\quad P(i-1,j)\omega^{D(x_i)} + P(i,j-1)\omega^{I(y_j)} \end{aligned}$$

Seeking $P(m,n)$ polynomial.
Still increased runtime and memory.

Fourier Transform Approach

Recap: Score Distribution via the Polynomial Approach

Coefficients of $P(m,n)$ are the score distribution. Use:

$$\begin{aligned} P(i,j) &= P(i-1,j-1)\omega^{M(x_i,y_j)} + \\ & \quad P(i-1,j)\omega^{D(x_i)} + P(i,j-1)\omega^{I(y_j)} \end{aligned}$$

Score Distribution via Fourier Transforms

Can recover coefficients of $P(m,n)$ with via its valuation at sufficiently many points. Its value for a fixed ω is from:

$$\begin{aligned} C(i,j) &= C(i-1,j-1)\omega^{M(x_i,y_j)} + \\ & \quad C(i-1,j)\omega^{D(x_i)} + C(i,j-1)\omega^{I(y_j)} \end{aligned}$$

Coefficients recovery is efficient via Fourier Transform, so let $\{\omega_0,\ldots,\omega_{r-1}\}$ be the rth roots of unity. Complex numbers.

Fourier Transform Approach

function ComputeScoreDistribution
for $k \in \{0, \ldots, r-1\}$
    $\omega = \cos(2\pi k/r) + i \sin(2\pi k/r)$
    $f(k) =$ BackgroundExec(CalcF(ω))
WaitForBackgroundProcesses
return FourierTransform(f)

function CalcF(ComplexNumber ω)
for $i \in \{0, \ldots, m\}$
    for $j \in \{0, \ldots, n\}$
        $C(i,j) = C(i-1,j-1)\omega^{M(x_i,y_j)} + C(i-1,j)\omega^{D(x_i)} + C(i,j-1)\omega^{I(y_j)}$
return $C(m,n)$

- Serial algorithm has original memory requirement.
- Parallel algorithm has (nearly) original runtime.

Results

The Result Is General

The approach applies quite generally to dynamic programming algorithms that compute an integer score.
Computing the score distribution:

- Serial algorithm has original memory requirement.
- Parallel algorithm has (nearly) original runtime.

An Example: Alignment Credibility

A tack in focus: Credibility = Bayesian Confidence.

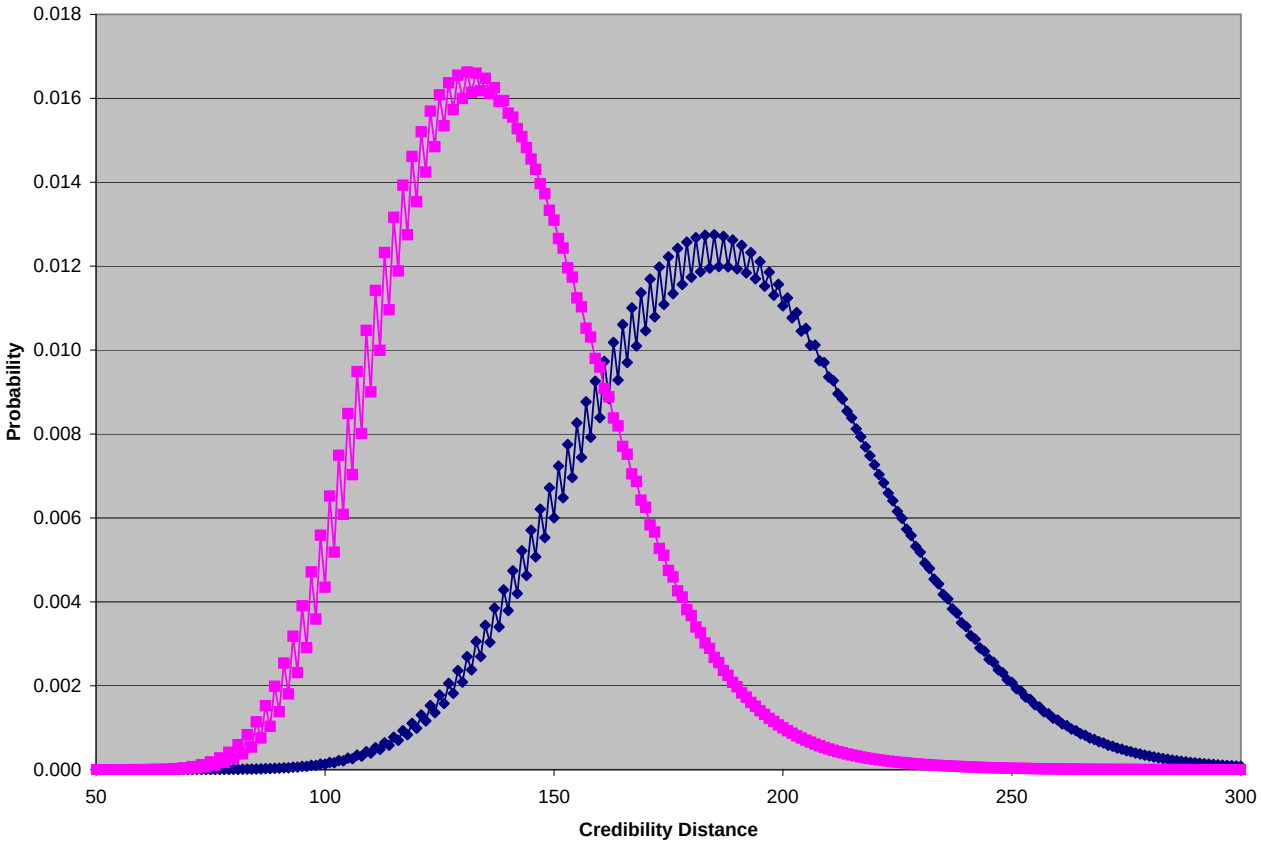
Number of Pairing Differences from Reference Alignment

- For sequences x and y, set probability of an alignment A with score $s(x, y, A)$ to be:

$$\Pr[A|x, y] \propto \exp(\lambda s(x, y, A))$$

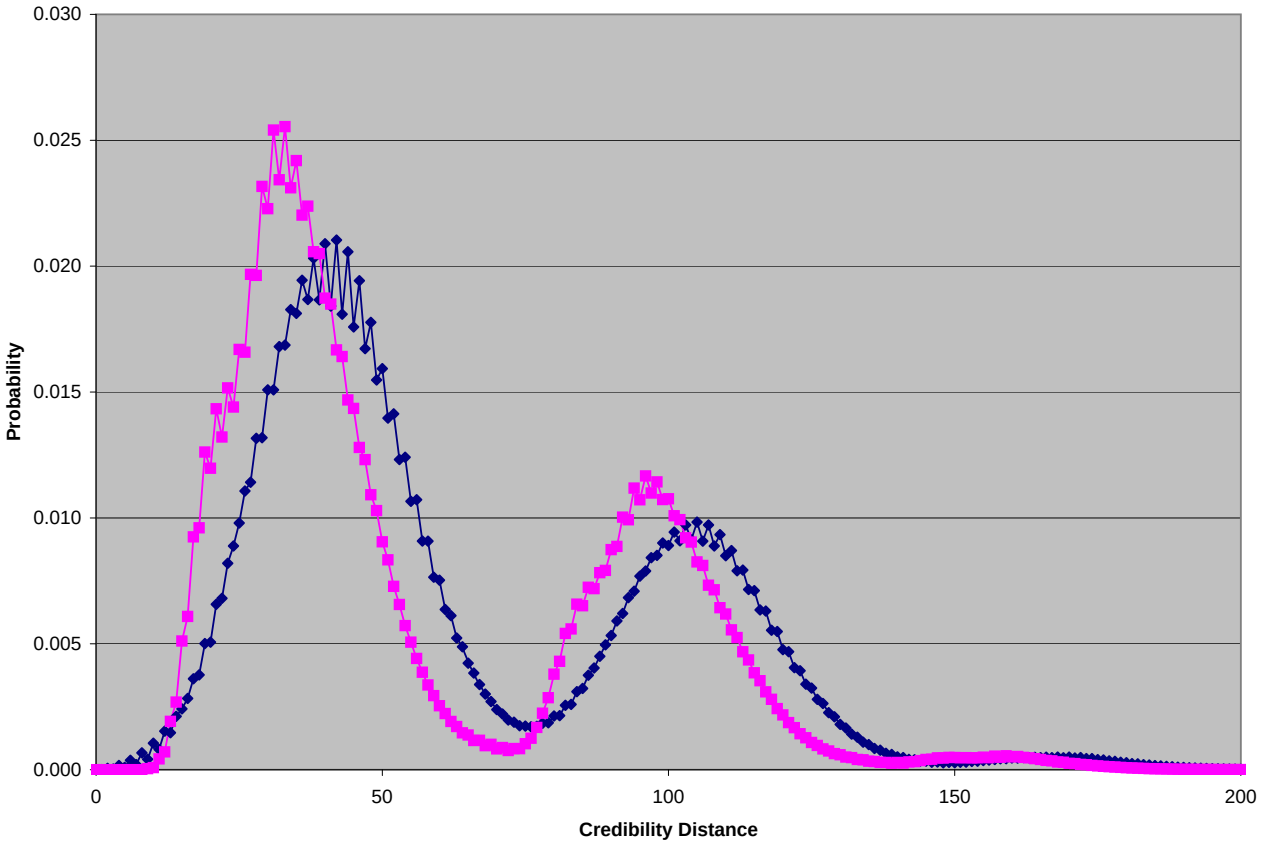
  for some parameter $\lambda > 0$.
- An alignment chosen at random from $\Pr[A|x, y]$ may differ from a reference alignment. We can exactly calculate the distribution of the number of pairing differences.
- Can calculate x% credibility. $x = 50\%, 95\%, 99\%$, etc.

Number of Pairing Differences: Centroid Vs. Optimal



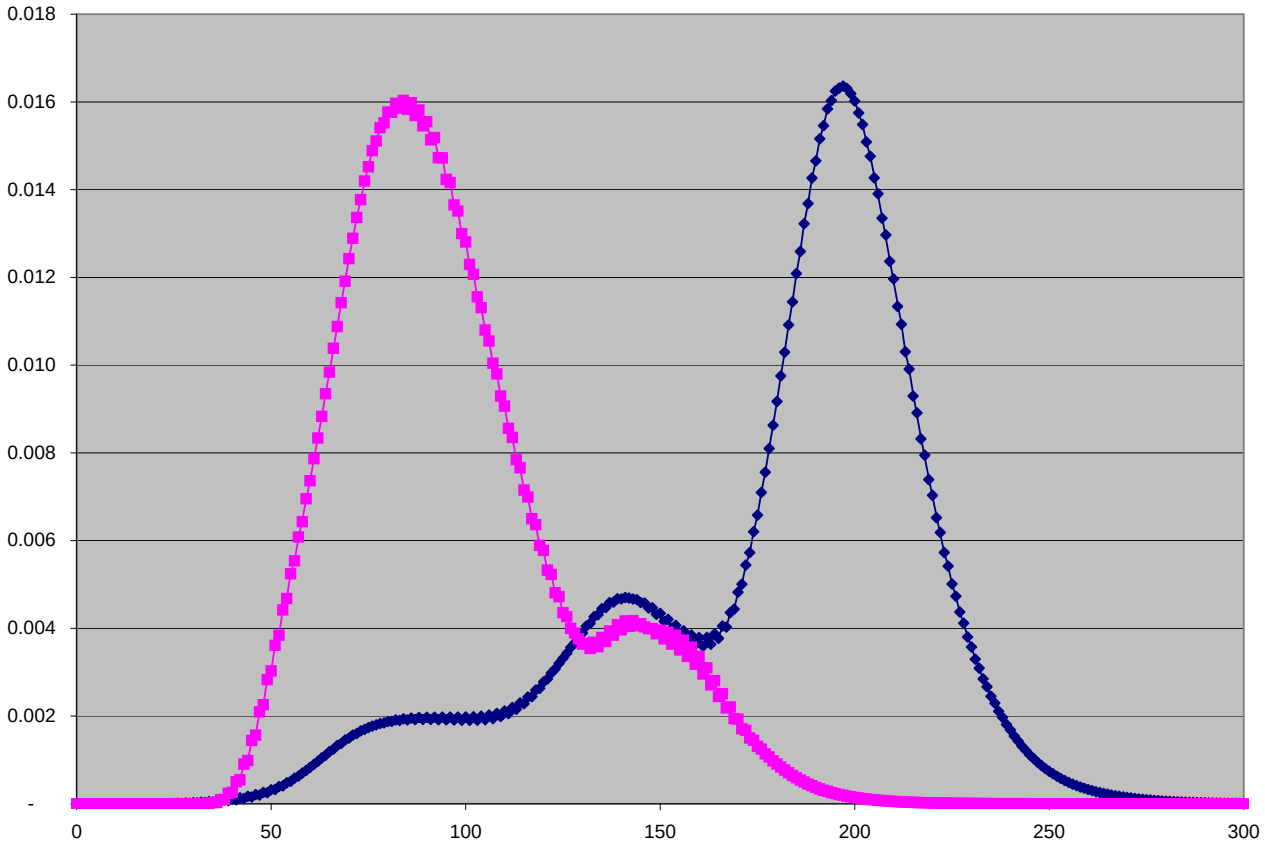
Human-rodent pairwise alignment, example #1.

Number of Pairing Differences: Bimodal



Human-rodent pairwise alignment, example #2.

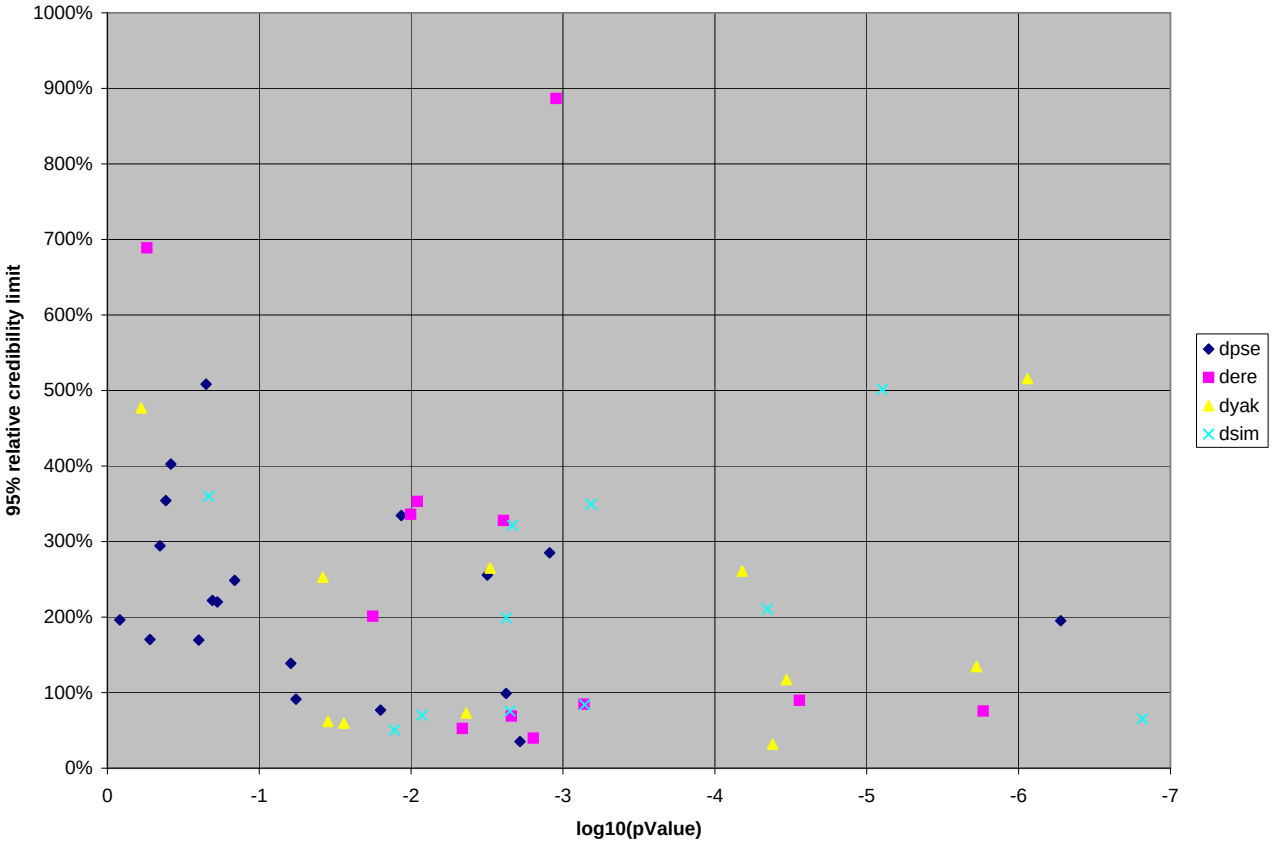
Number of Pairing Differences: Rich Structure



Human-rodent pairwise alignment, example #3.

Credibility Vs. Weak Statistical Significance

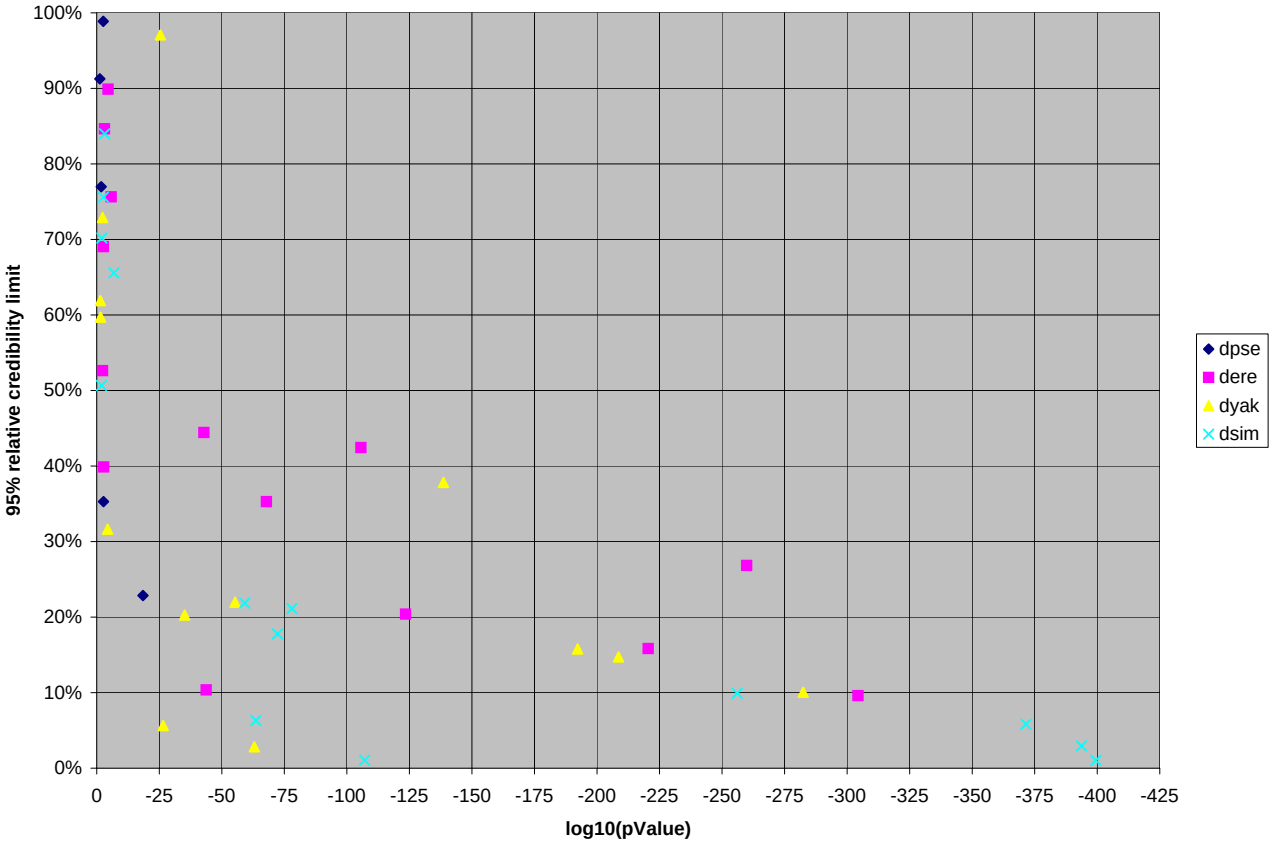
Credibility when statistical significance is relatively weak.



Drosophila melanogaster (fly) pairwise alignments.

Credibility Vs. Strong Statistical Significance

Credibility when statistical significance is strong.



Drosophila melanogaster (fly) pairwise alignments.

Conclusions

Take-Home Points

- **Background:** Often dynamic programming algorithms seek an optimal integer score, but entire distribution of scores can be of interest.
- **Results:** Three ways to compute distribution of scores. Applied to pairwise alignment of DNA sequences.
- **Conclusions:** Serial algorithm has no increased memory requirement. Highly parallelizable. Credibility $\neq$ statistical significance.

leen.cs@rpi@edu

`http://www.rpi.edu/~newbel/publications/NewbergRPICS2009.pdf`

`http://dx.doi.org/10.1089/cmb.2008.0137`