

Correctness and Performance Charts

This version of the document is dated 2022-11-07.

The following charts show the correctness of many of the algorithms in "**Bernoulli Factory Algorithms**" and show their performance in terms of the number of bits they use on average. For each algorithm, and for each of 100 λ values evenly spaced from 0.0001 to 0.9999:

- 500 runs of the algorithm were done. Then...
- The number of bits used by the runs were averaged, as were the return values of the runs (since the return value is either 0 or 1, the mean return value will be in the interval $[0, 1]$). The number of bits used included the number of bits used to produce each coin flip, assuming the coin flip procedure for λ was generated using the `Bernoulli#coin()` method in *bernoulli.py*, which produces that probability in an optimal or near-optimal way.

For each algorithm, if a single run was detected to use more than 5000 bits for a given λ , the entire data point for that λ was suppressed in the charts below.

In addition, for each algorithm, a chart appears showing the minimum number of input coin flips that any fast Bernoulli factory algorithm will need on average to simulate the given function, based on work by Mendo (2019)[¹]. Note that some functions require a growing number of coin flips as λ approaches 0 or 1. Note that for the 2014, 2016, and 2019 algorithms—

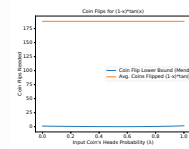
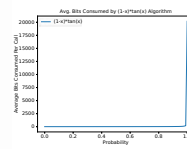
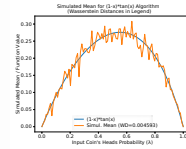
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

Points with invalid ϵ values were suppressed. For the low-mean algorithm, an m of $\max(0.49999, x*c*1.02)$ was used unless noted otherwise.

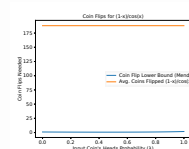
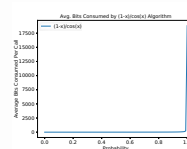
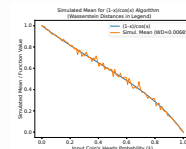
0.1 The Charts

Algorithm	Simulated Mean	Average Bits Consumed	Coin Flips
-----------	----------------	-----------------------	------------

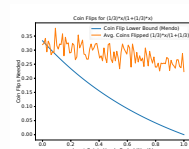
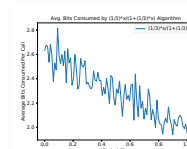
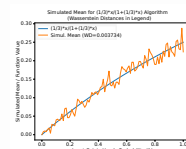
$$(1-x)*\tan(x)$$



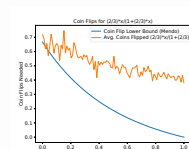
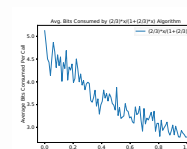
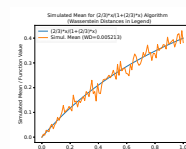
$$(1-x)/\cos(x)$$



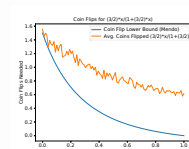
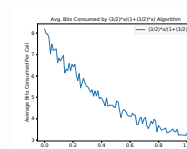
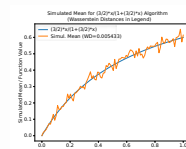
$$(1/3)*x/(1+(1/3)*x)$$



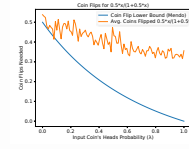
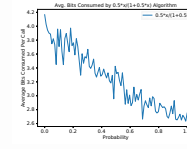
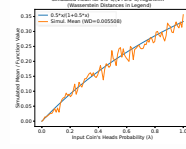
$$(2/3)*x/(1+(2/3)*x)$$



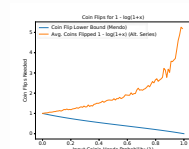
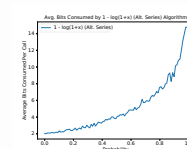
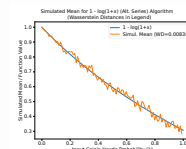
$$(3/2)*x/(1+(3/2)*x)$$



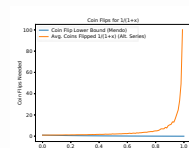
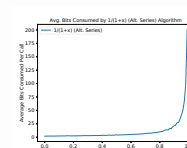
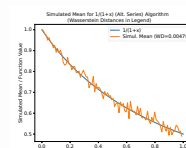
$$0.5*x/(1+0.5*x)$$



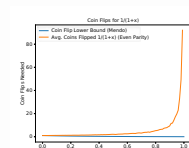
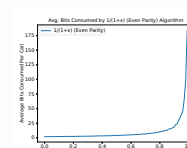
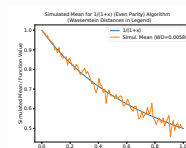
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



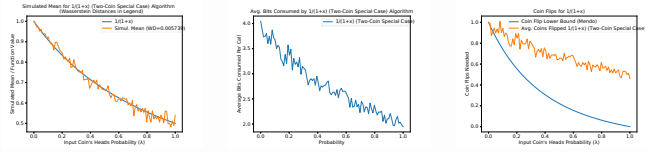
$$1/(1+x) \text{ (Alt. Series)}$$



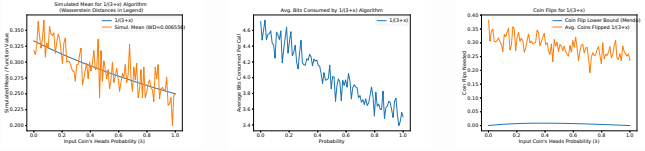
$$1/(1+x) \text{ (Even Parity)}$$



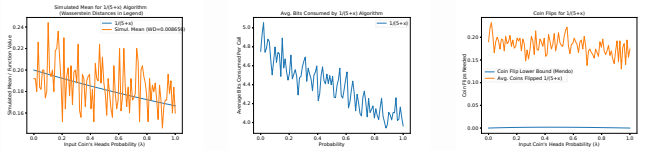
1/(1+x) (Two-Coin Special Case)



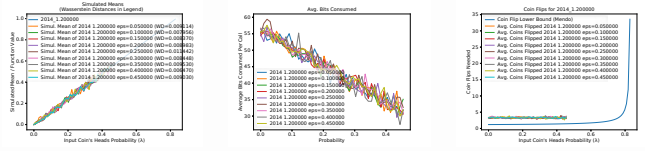
1/(3+x)



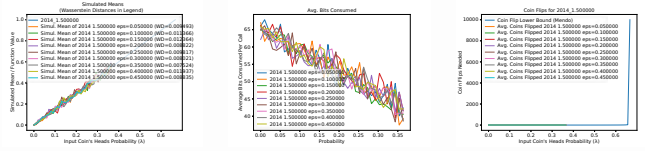
1/(5+x)



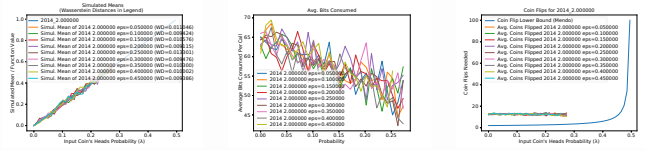
2014 1.200000
eps=0.050000



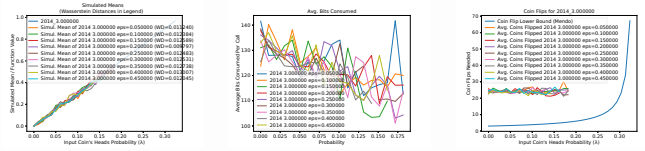
2014 1.500000
eps=0.050000



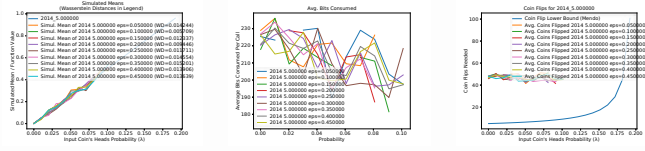
2014 2.000000
eps=0.050000



2014 3.000000
eps=0.050000



2014 5.000000
eps=0.050000



2014 Add. x+0.1

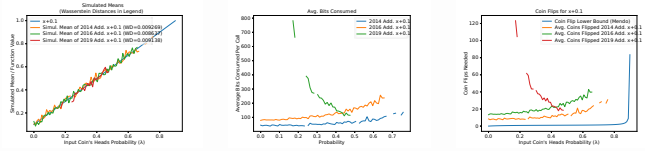


Figure 1 consists of three subplots. The left subplot, titled 'Disaggregated Rain (Observations and Proposed Method)', shows True Rain (mm) on the y-axis (0.0 to 1.0) versus Predicted Rain (mm) on the x-axis (0.0 to 1.0). It includes data for 2014 (blue), 2015 (orange), and 2016 (green), along with a 1:1 line. The middle subplot, titled 'Avg. RMSE (mm) vs. Correlation Coefficient', shows Average RMSE (mm) on the y-axis (0.0 to 0.5) versus Correlation Coefficient on the x-axis (0.0 to 1.0). It includes data for 2014 (blue), 2015 (orange), and 2016 (green), along with a 1:1 line. The right subplot, titled 'Error Rate (%) vs. Correlation Coefficient', shows Error Rate (%) on the y-axis (0 to 100) versus Correlation Coefficient on the x-axis (0.0 to 1.0). It includes data for 2014 (blue), 2015 (orange), and 2016 (green), along with a 1:1 line.

Figure 1 consists of three subplots comparing the proposed method (CUPR) with state-of-the-art methods (CUPR-Learner, CUPR-Ensemble, and CUPR-Boost) across different years (2014, 2015, 2016, 2017, 2018, 2019) for AUC values ranging from 0.6 to 0.9.

Top Left Plot: Stratified Mean AUC (std) vs AUC

- Y-axis:** Stratified Mean AUC (std) ranging from 0.6 to 0.9.
- X-axis:** AUC ranging from 0.6 to 0.9.
- Legend:**
 - 2014, AUC=0.6
 - 2014, AUC=0.7
 - 2014, AUC=0.8
 - 2014, AUC=0.9
 - 2015, AUC=0.6
 - 2015, AUC=0.7
 - 2015, AUC=0.8
 - 2015, AUC=0.9
 - 2016, AUC=0.6
 - 2016, AUC=0.7
 - 2016, AUC=0.8
 - 2016, AUC=0.9
 - 2017, AUC=0.6
 - 2017, AUC=0.7
 - 2017, AUC=0.8
 - 2017, AUC=0.9
 - 2018, AUC=0.6
 - 2018, AUC=0.7
 - 2018, AUC=0.8
 - 2018, AUC=0.9
 - 2019, AUC=0.6
 - 2019, AUC=0.7
 - 2019, AUC=0.8
 - 2019, AUC=0.9

Top Right Plot: Average ROC Curves for AUC

- Y-axis:** Average ROC Curves for AUC ranging from 0.6 to 0.9.
- X-axis:** AUC ranging from 0.6 to 0.9.
- Legend:**
 - 2014, AUC=0.6
 - 2014, AUC=0.7
 - 2014, AUC=0.8
 - 2014, AUC=0.9
 - 2015, AUC=0.6
 - 2015, AUC=0.7
 - 2015, AUC=0.8
 - 2015, AUC=0.9
 - 2016, AUC=0.6
 - 2016, AUC=0.7
 - 2016, AUC=0.8
 - 2016, AUC=0.9
 - 2017, AUC=0.6
 - 2017, AUC=0.7
 - 2017, AUC=0.8
 - 2017, AUC=0.9
 - 2018, AUC=0.6
 - 2018, AUC=0.7
 - 2018, AUC=0.8
 - 2018, AUC=0.9
 - 2019, AUC=0.6
 - 2019, AUC=0.7
 - 2019, AUC=0.8
 - 2019, AUC=0.9

Bottom Plot: CUPR Error for AUC=0.5

- Y-axis:** CUPR Error for AUC=0.5 ranging from 0.0 to 0.5.
- X-axis:** AUC ranging from 0.6 to 0.9.
- Legend:**
 - CUPR-Learner: Boosted (green line)
 - CUPR-Learner: Ensemble (orange line)
 - CUPR-Learner: Boosted (blue line)
 - CUPR-Ensemble: Boosted (red line)
 - CUPR-Ensemble: Ensemble (yellow line)
 - CUPR-Ensemble: Boosted (purple line)
 - CUPR-Boost: Boosted (brown line)
 - CUPR-Boost: Ensemble (pink line)
 - CUPR-Boost: Boosted (grey line)

[illegible][illegible]

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics against the 'Input CoV's Weights Probability' on the x-axis (ranging from 0.0 to 0.4).

Subplot (a) is titled 'Resemblance Distance vs. Input CoV's Weights Probability'. The y-axis is 'Resemblance Distance (Cost)' ranging from 0.0 to 1.0. It shows a linear relationship between the input probability and the resemblance distance for various cost functions. The legend includes:

- Cost: $\varphi_1(x)$ (blue line)
- Cost: $\varphi_2(x)$ (orange line)
- Cost: $\varphi_3(x)$ (green line)
- Cost: $\varphi_4(x)$ (red line)
- Cost: $\varphi_5(x)$ (purple line)
- Cost: $\varphi_6(x)$ (brown line)
- Cost: $\varphi_7(x)$ (pink line)
- Cost: $\varphi_8(x)$ (grey line)
- Cost: $\varphi_9(x)$ (light blue line)
- Cost: $\varphi_{10}(x)$ (light green line)
- Cost: $\varphi_{11}(x)$ (light orange line)
- Cost: $\varphi_{12}(x)$ (light red line)
- Cost: $\varphi_{13}(x)$ (light purple line)
- Cost: $\varphi_{14}(x)$ (light brown line)
- Cost: $\varphi_{15}(x)$ (light pink line)
- Cost: $\varphi_{16}(x)$ (light grey line)
- Cost: $\varphi_{17}(x)$ (light light blue line)
- Cost: $\varphi_{18}(x)$ (light light green line)
- Cost: $\varphi_{19}(x)$ (light light orange line)
- Cost: $\varphi_{20}(x)$ (light light red line)
- Cost: $\varphi_{21}(x)$ (light light purple line)
- Cost: $\varphi_{22}(x)$ (light light brown line)
- Cost: $\varphi_{23}(x)$ (light light pink line)
- Cost: $\varphi_{24}(x)$ (light light grey line)
- Cost: $\varphi_{25}(x)$ (light light light blue line)
- Cost: $\varphi_{26}(x)$ (light light light green line)
- Cost: $\varphi_{27}(x)$ (light light light orange line)
- Cost: $\varphi_{28}(x)$ (light light light red line)
- Cost: $\varphi_{29}(x)$ (light light light purple line)
- Cost: $\varphi_{30}(x)$ (light light light brown line)
- Cost: $\varphi_{31}(x)$ (light light light pink line)
- Cost: $\varphi_{32}(x)$ (light light light grey line)
- Cost: $\varphi_{33}(x)$ (light light light light blue line)
- Cost: $\varphi_{34}(x)$ (light light light light green line)
- Cost: $\varphi_{35}(x)$ (light light light light orange line)
- Cost: $\varphi_{36}(x)$ (light light light light red line)
- Cost: $\varphi_{37}(x)$ (light light light light purple line)
- Cost: $\varphi_{38}(x)$ (light light light light brown line)
- Cost: $\varphi_{39}(x)$ (light light light light pink line)
- Cost: $\varphi_{40}(x)$ (light light light light grey line)
- Cost: $\varphi_{41}(x)$ (light light light light light blue line)
- Cost: $\varphi_{42}(x)$ (light light light light light green line)
- Cost: $\varphi_{43}(x)$ (light light light light light orange line)
- Cost: $\varphi_{44}(x)$ (light light light light light red line)
- Cost: $\varphi_{45}(x)$ (light light light light light purple line)
- Cost: $\varphi_{46}(x)$ (light light light light light brown line)
- Cost: $\varphi_{47}(x)$ (light light light light light pink line)
- Cost: $\varphi_{48}(x)$ (light light light light light grey line)
- Cost: $\varphi_{49}(x)$ (light light light light light light blue line)
- Cost: $\varphi_{50}(x)$ (light light light light light light green line)
- Cost: $\varphi_{51}(x)$ (light light light light light light orange line)
- Cost: $\varphi_{52}(x)$ (light light light light light light red line)
- Cost: $\varphi_{53}(x)$ (light light light light light light purple line)
- Cost: $\varphi_{54}(x)$ (light light light light light light brown line)
- Cost: $\varphi_{55}(x)$ (light light light light light light pink line)
- Cost: $\varphi_{56}(x)$ (light light light light light light grey line)
- Cost: $\varphi_{57}(x)$ (light light light light light light light blue line)
- Cost: $\varphi_{58}(x)$ (light light light light light light light green line)
- Cost: $\varphi_{59}(x)$ (light light light light light light light orange line)
- Cost: $\varphi_{60}(x)$ (light light light light light light light red line)
- Cost: $\varphi_{61}(x)$ (light light light light light light light purple line)
- Cost: $\varphi_{62}(x)$ (light light light light light light light brown line)
- Cost: $\varphi_{63}(x)$ (light light light light light light light pink line)
- Cost: $\varphi_{64}(x)$ (light light light light light light light grey line)
- Cost: $\varphi_{65}(x)$ (light light light light light light light light blue line)
- Cost: $\varphi_{66}(x)$ (light light light light light light light light green line)
- Cost: $\varphi_{67}(x)$ (light light light light light light light light orange line)
- Cost: $\varphi_{68}(x)$ (light light light light light light light light red line)
- Cost: $\varphi_{69}(x)$ (light light light light light light light light purple line)
- Cost: $\varphi_{70}(x)$ (light light light light light light light light brown line)
- Cost: $\varphi_{71}(x)$ (light light light light light light light light pink line)
- Cost: $\varphi_{72}(x)$ (light light light light light light light light grey line)
- Cost: $\varphi_{73}(x)$ (light light light light light light light light light blue line)
- Cost: $\varphi_{74}(x)$ (light light light light light light light light light green line)
- Cost: $\varphi_{75}(x)$ (light light light light light light light light light orange line)
- Cost: $\varphi_{76}(x)$ (light light light light light light light light light red line)
- Cost: $\varphi_{77}(x)$ (light light light light light light light light light purple line)
- Cost: $\varphi_{78}(x)$ (light light light light light light light light light brown line)
- Cost: $\varphi_{79}(x)$ (light light light light light light light light light pink line)
- Cost: $\varphi_{80}(x)$ (light light light light light light light light light grey line)
- Cost: $\varphi_{81}(x)$ (light light light light light light light light light light blue line)
- Cost: $\varphi_{82}(x)$ (light light light light light light light light light light green line)
- Cost: $\varphi_{83}(x)$ (light light light light light light light light light light orange line)
- Cost: $\varphi_{84}(x)$ (light light light light light light light light light light red line)
- Cost: $\varphi_{85}(x)$ (light light light light light light light light light light purple line)
- Cost: $\varphi_{86}(x)$ (light light light light light light light light light light brown line)
- Cost: $\varphi_{87}(x)$ (light light light light light light light light light light pink line)
- Cost: $\varphi_{88}(x)$ (light light light light light light light light light light grey line)
- Cost: $\varphi_{89}(x)$ (light light light light light light light light light light light blue line)
- Cost: $\varphi_{90}(x)$ (light light light light light light light light light light light green line)
- Cost: $\varphi_{91}(x)$ (light light light light light light light light light light light orange line)
- Cost: $\varphi_{92}(x)$ (light light light light light light light light light light light red line)
- Cost: $\varphi_{93}(x)$ (light light light light light light light light light light light purple line)
- Cost: $\varphi_{94}(x)$ (light light light light light light light light light light light brown line)
- Cost: $\varphi_{95}(x)$ (light light light light light light light light light light light pink line)
- Cost: $\varphi_{96}(x)$ (light light light light light light light light light light light grey line)
- Cost: $\varphi_{97}(x)$ (light light light light light light light light light light light light blue line)
- Cost: $\varphi_{98}(x)$ (light light light light light light light light light light light light green line)
- Cost: $\varphi_{99}(x)$ (light light light light light light light light light light light light orange line)
- Cost: $\varphi_{100}(x)$ (light light light light light light light light light light light light red line)

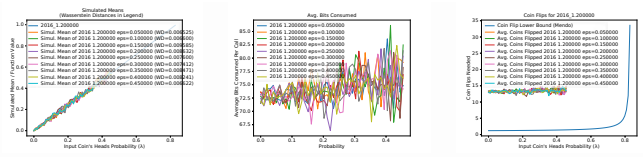
Subplot (b) is titled 'Avg. Miss Classification vs. Probability'. The y-axis is 'Average Miss Classification' ranging from 0.0 to 1.0. It shows the average miss classification rate for various cost functions. The legend includes:

- Cost: $\varphi_1(x)$ (blue line)
- Cost: $\varphi_2(x)$ (orange line)
- Cost: $\varphi_3(x)$ (green line)
- Cost: $\varphi_4(x)$ (red line)
- Cost: $\varphi_5(x)$ (purple line)
- Cost: $\varphi_6(x)$ (brown line)
- Cost: $\varphi_7(x)$ (pink line)
- Cost: $\varphi_8(x)$ (grey line)
- Cost: $\varphi_9(x)$ (light blue line)
- Cost: $\varphi_{10}(x)$ (light green line)
- Cost: $\varphi_{11}(x)$ (light orange line)
- Cost: $\varphi_{12}(x)$ (light red line)
- Cost: $\varphi_{13}(x)$ (light purple line)
- Cost: $\varphi_{14}(x)$ (light brown line)
- Cost: $\varphi_{15}(x)$ (light pink line)
- Cost: $\varphi_{16}(x)$ (light grey line)
- Cost: $\varphi_{17}(x)$ (light light blue line)
- Cost: $\varphi_{18}(x)$ (light light green line)
- Cost: $\varphi_{19}(x)$ (light light orange line)
- Cost: $\varphi_{20}(x)$ (light light red line)
- Cost: $\varphi_{21}(x)$ (light light purple line)
- Cost: $\varphi_{22}(x)$ (light light brown line)
- Cost: $\varphi_{23}(x)$ (light light pink line)
- Cost: $\varphi_{24}(x)$ (light light grey line)
- Cost: $\varphi_{25}(x)$ (light light light blue line)
- Cost: $\varphi_{26}(x)$ (light light light green line)
- Cost: $\varphi_{27}(x)$ (light light light orange line)
- Cost: $\varphi_{28}(x)$ (light light light red line)
- Cost: $\varphi_{29}(x)$ (light light light purple line)
- Cost: $\varphi_{30}(x)$ (light light light brown line)
- Cost: $\varphi_{31}(x)$ (light light light pink line)
- Cost: $\varphi_{32}(x)$ (light light light grey line)
- Cost: $\varphi_{33}(x)$ (light light light light blue line)
- Cost: $\varphi_{34}(x)$ (light light light light green line)
- Cost: $\varphi_{35}(x)$ (light light light light orange line

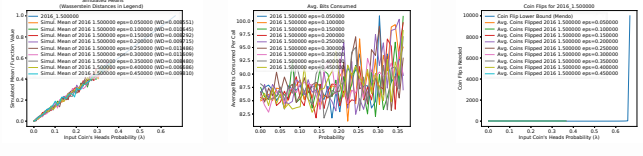
Figure 10 consists of three subplots. The left subplot shows the dry-film thickness (micrometers) versus the resin coat's weight percentage (%). The data points for various resin coats (4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100) are plotted, and a linear fit is shown. The middle subplot shows the average modulus (GPa) versus porosity for various resin coats (4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100). The right subplot shows the coating thickness (micrometers) versus the resin coat's weight percentage (%) for different curing conditions (Cure Time: 10 min, 20 min, 30 min, 40 min, 50 min, 60 min, 70 min, 80 min, 90 min, 100 min; Cure Temp: 40°C, 50°C, 60°C, 70°C, 80°C, 90°C, 100°C).

[illegible]

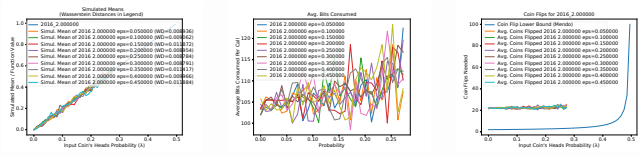
```
2016 1.200000
eps=0.050000
```



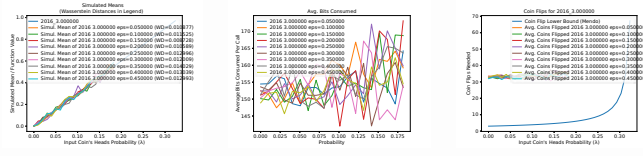
```
2016 1.500000
eps=0.050000
```



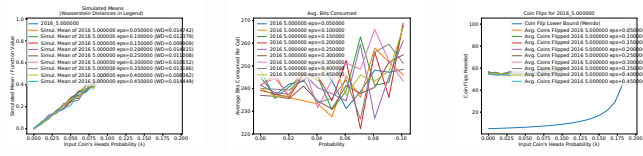
```
2016 2.000000
eps=0.050000
```



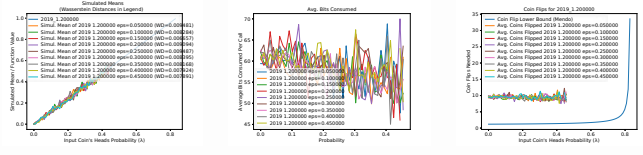
```
2016 3.000000
eps=0.050000
```



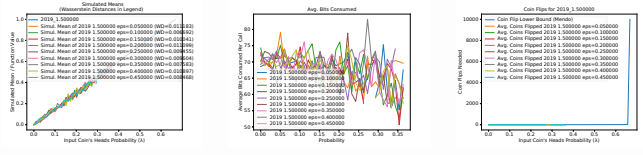
```
2016 5.000000
eps=0.050000
```



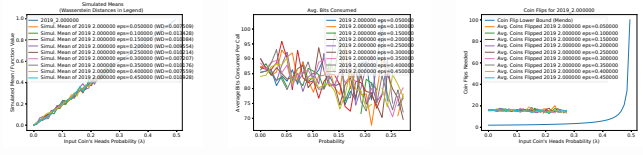
```
2019 1.200000
eps=0.050000
```



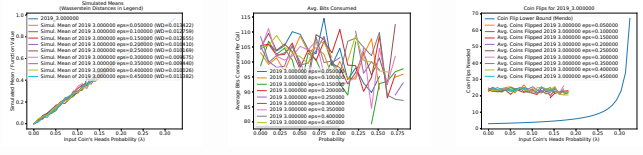
```
2019 1.500000
eps=0.050000
```



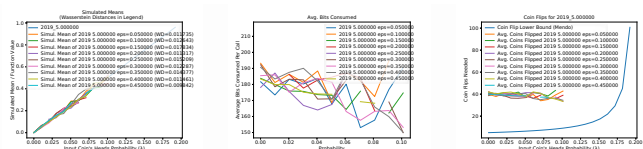
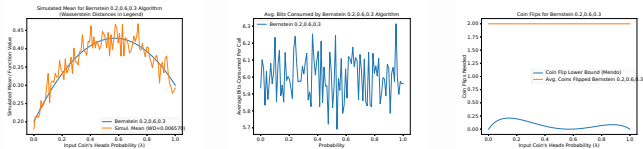
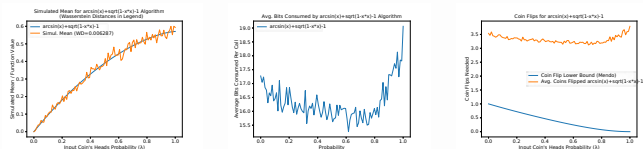
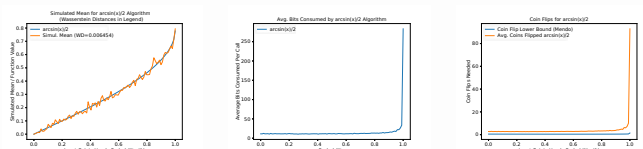
```
2019 2.000000
eps=0.050000
```



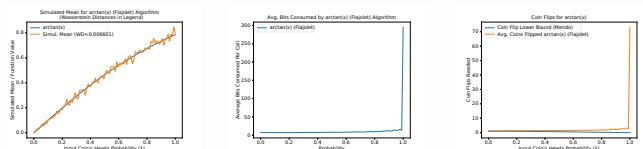
2019 3.000000
eps=0.050000



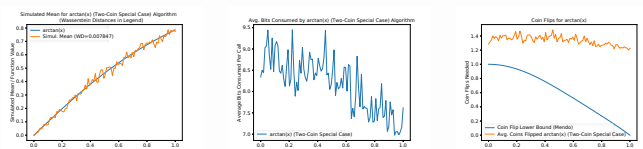
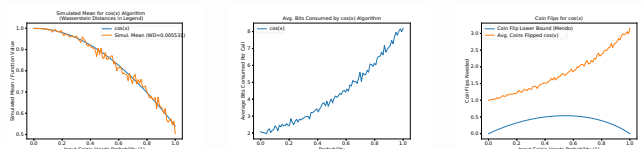
```
2019 5.000000
eps=0.050000
```

Bernstein
0.2,0.6,0.3
$$\arcsin(x) + \sqrt{1-x^2} - 1$$
 $\arcsin(x)/2$ 

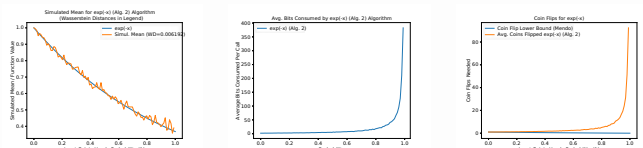
arctan(x)
(Flajolet)



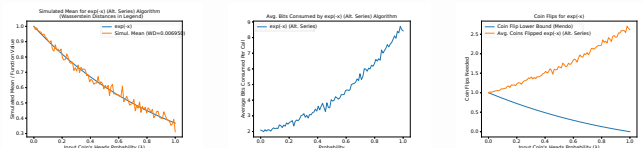
arctan(x) (Two-Coin Special Case)

 $\cos(x)$ 

$\exp(-x)$ (Alg. 2)



exp(-x) (Alt.
Series)



Estimated Mean for each of (Empirical) Algorithms
 Estimated Mean for each of (Empirical) Algorithms
 Exact Mean (0.001-0.005) Prop. Mean (0.001-0.005)

Average Coefficient (beta)
 Exact (0.001-0.005) Prop. (0.001-0.005)

Gini Index (times) Risk
 Exact (0.001-0.005) Prop. (0.001-0.005)

Figure 10 consists of three subplots. The left subplot, titled 'Scalability of the proposed algorithm', shows the 'Scalability of the proposed algorithm' on the y-axis (ranging from 0.0 to 1.0) versus 'Input Data Size (Number of Nodes)' on the x-axis (ranging from 0.0 to 1.0). It compares 'Proposed' (blue line) and 'Baseline' (orange line) algorithms. Both show a decreasing trend, with the proposed algorithm maintaining higher scalability at larger input sizes. The middle subplot, titled 'Avg. Bits Consumed by the proposed algorithm', shows 'Avg. Bits Consumed (Bits)' on the y-axis (ranging from 0 to 10) versus 'Input Data Size (Number of Nodes)' on the x-axis (ranging from 0.0 to 1.0). It compares 'Proposed' (blue line) and 'Baseline' (orange line) algorithms. Both show an increasing trend, with the proposed algorithm consuming fewer bits. The right subplot, titled 'Gap Error for input data size', shows 'Gap Error (Error)' on the y-axis (ranging from 0.0 to 0.3) versus 'Input Data Size (Number of Nodes)' on the x-axis (ranging from 0.0 to 1.0). It compares 'Proposed' (blue line) and 'Baseline' (orange line) algorithms. Both show an increasing trend, with the proposed algorithm having a lower gap error.

Figure 1 consists of three subplots comparing the proposed algorithm with the Pigeon-inspired algorithm. The left subplot shows the 'Serialized Mean for $\log_2(x+1)$ (Pigeon)' and 'Serialized Mean for $\log_2(x+1)$ (Proposed)' algorithms. The proposed algorithm shows a lower mean than the Pigeon algorithm. The middle subplot shows the 'log2(x+1) (Pigeon)' and 'log2(x+1) (Proposed)' algorithms. The proposed algorithm shows a lower mean than the Pigeon algorithm. The right subplot shows the 'Cost Ratio for $\log_2(x+1)$ ' for the 'Cost-Flip Greedy' and 'Avg. Cost Pigeon log2(x+1) (Pigeon)' algorithms. The proposed algorithm shows a lower cost ratio than the Pigeon algorithm.

Figure 1 consists of three subplots comparing the proposed algorithm with the state-of-the-art. The left subplot shows the Graded Mean Error (log10) versus log10(1/epsilon) for two values of log10(1/epsilon): 0.000001 (blue line) and 0.0000001 (orange line). The middle subplot shows the Average Work Computed by log10(1/epsilon) versus log10(1/epsilon) for the same two values. The right subplot shows the Cost (log) versus log10(1/epsilon) for the same two values. The x-axis for all plots is log10(1/epsilon) ranging from -0.6 to 1.6. The y-axis for the left plot is Graded Mean Error (log10) ranging from 0.0 to 0.4. The y-axis for the middle plot is Average Work Computed by log10(1/epsilon) ranging from 0.0 to 1.0. The y-axis for the right plot is Cost (log) ranging from 0.0 to 1.8.

Figure 1 consists of three subplots labeled (a), (b), and (c).
 Subplot (a) is titled 'Scalability: Disabled flow capacity (kA) vs. Number of nodes'. The x-axis is 'Number of nodes' ranging from 0.0 to 1.0. The y-axis is 'Disabled flow capacity (kA)' ranging from 0.0 to 1.0. It shows two curves: 'points 1-10' (blue line with circles) and 'points 100-1000' (orange line with circles). Both curves show an increasing trend, with the orange curve being slightly higher than the blue curve.
 Subplot (b) is titled 'Convergence: Average flow capacity (kA) vs. Iteration number'. The x-axis is 'Iteration number' ranging from 0.0 to 1.0. The y-axis is 'Average flow capacity (kA)' ranging from 0.0 to 1000.0. It shows two curves: 'points 1-10' (blue line with circles) and 'points 100-1000' (orange line with circles). Both curves show a rapid decrease in capacity, stabilizing near zero after approximately 0.2 iterations.
 Subplot (c) is titled 'Error: Error (kA) vs. Iteration number'. The x-axis is 'Iteration number' ranging from 0.0 to 1.0. The y-axis is 'Error (kA)' ranging from 0 to 400. It shows two curves: 'points 1-10' (blue line with circles) and 'points 100-1000' (orange line with circles). Both curves show a rapid decrease in error, stabilizing near zero after approximately 0.2 iterations.

Figure 10 consists of three subplots comparing the performance of the proposed algorithm (grey line) with the state-of-the-art algorithm (orange line) for the $p=213$ case.

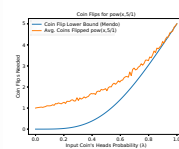
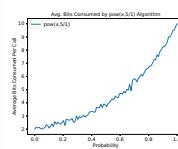
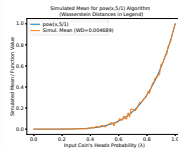
- Left Plot:** Dryad Error ($1/2 ||x||_2^2$) vs. Signal-to-noise ratio (SNR). The x-axis ranges from 0.0 to 1.0, and the y-axis ranges from 0.0 to 1.0. The legend indicates: grey line for $\text{error}_{\text{D}}(213)$ and orange line for $\text{error}_{\text{D}}(\text{Hedge}) (0.000308)$. Both lines show an increasing trend, with the grey line generally higher than the orange line.
- Middle Plot:** Average #Covariates vs. SNR. The x-axis ranges from 0.0 to 1.0, and the y-axis ranges from 0 to 6. The legend indicates: grey line for $\text{Avg. Size Computed by } \text{p}(\text{p}=213)$ and orange line for $\text{Avg. Size } (213)$. The grey line is a constant horizontal line at 213, while the orange line shows a noisy, increasing trend.
- Right Plot:** Case File Loss (RMSE) vs. Signal-to-noise ratio (SNR). The x-axis ranges from 0.0 to 1.0, and the y-axis ranges from 0.00 to 2.00. The legend indicates: grey line for $\text{Case File Loss (RMSE) Proposed}$ and orange line for $\text{Avg. Case File Loss}$. Both lines show an increasing trend, with the grey line generally higher than the orange line.

Figure 1 consists of three subplots labeled (a), (b), and (c), comparing the performance of the proposed algorithm (points 3,4) against a baseline (points 1,2).

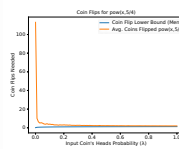
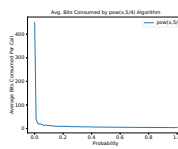
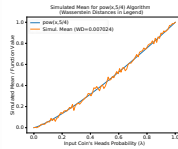
- (a) Standard Error vs. Capital Asset Turnover Ratio:** The y-axis is 'Standard Error (1000000000)' ranging from 0.0 to 1.0. The x-axis is 'Capital Asset Turnover Ratio (%)' ranging from 0.0 to 1.0. The 'points 3,4' series (blue line) shows a lower standard error than the 'points 1,2' series (orange line) across the range of ratios.
- (b) Avg. Bits Consumed vs. Precision:** The y-axis is 'Avg. Bits Consumed (1000000)' ranging from 0 to 100. The x-axis is 'Precision' ranging from 0.0 to 1.0. The 'points 3,4' series (blue line) shows a much lower average number of bits consumed than the 'points 1,2' series (orange line) for most precision values.
- (c) Cost Functions vs. Capital Asset Turnover Ratio:** The y-axis is 'Cost Functions' ranging from 0 to 1.4. The x-axis is 'Capital Asset Turnover Ratio (%)' ranging from 0.0 to 1.0. The 'Cost Funtion Based (points 3,4)' series (blue line) shows a lower cost function value than the 'Cost Funtion Based (points 1,2)' series (orange line) for most ratios.

Figure 1 consists of two subplots. Subplot (a) is a Receiver Operating Characteristic (ROC) curve showing the Simulated Mean Fraction Rate (Y-axis, 0.0 to 1.0) versus the True Fraction Rate (X-axis, 0.0 to 1.0). It compares three algorithms: pnorm (blue line), K5 (orange line), and pnorm + K5 (red line). The pnorm + K5 curve is the highest, followed by K5, and then pnorm. Subplot (b) is a line graph showing Average Bits per Symbol (Y-axis, 0.0 to 10.0) versus Input CS's Mean Probability (X-axis, 0.0 to 1.0). It compares pnorm (blue line) and pnorm + K5 (red line). The pnorm + K5 curve is significantly lower than the pnorm curve, indicating better performance in terms of average bits per symbol.

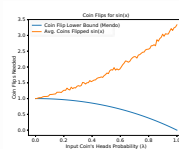
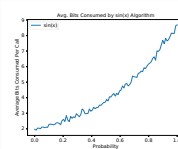
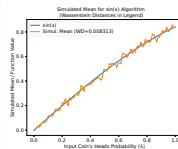
pow(x,5/1)



pow(x,5/4)



sin(x)



sqrt(x)

