

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—

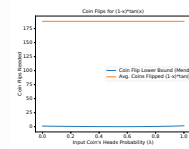
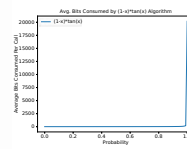
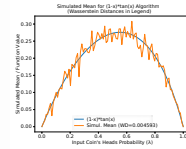
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

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.

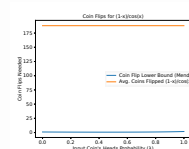
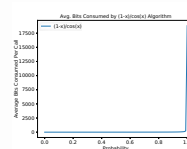
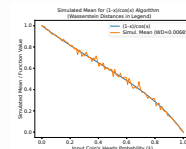
0.1 The Charts

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

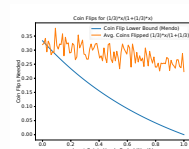
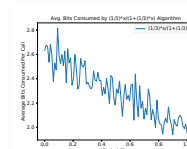
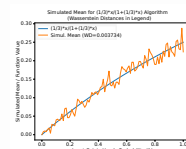
$$(1-x)*\tan(x)$$



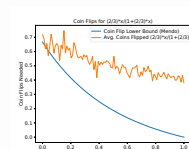
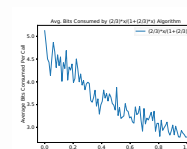
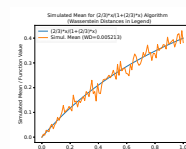
$$(1-x)/\cos(x)$$



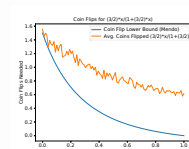
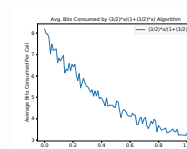
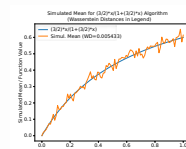
$$(1/3)*x/(1+(1/3)*x)$$



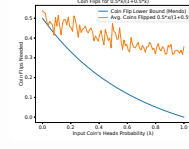
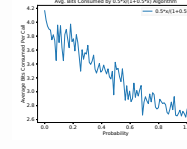
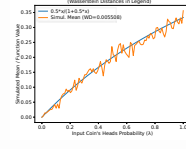
$$(2/3)*x/(1+(2/3)*x)$$



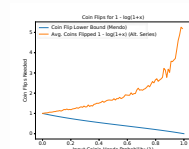
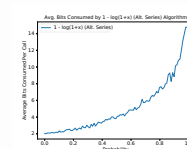
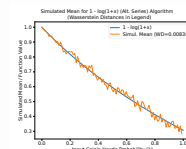
$$(3/2)*x/(1+(3/2)*x)$$



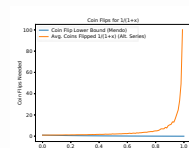
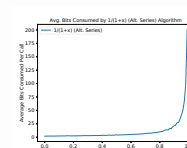
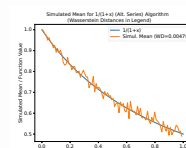
$$0.5*x/(1+0.5*x)$$



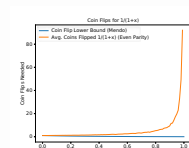
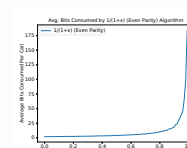
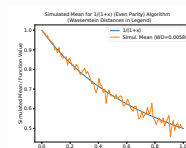
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



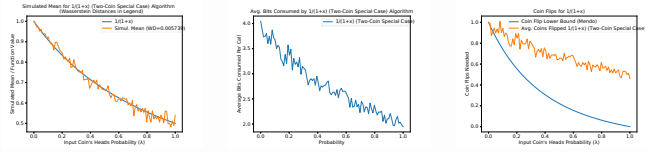
$$1/(1+x) \text{ (Alt. Series)}$$



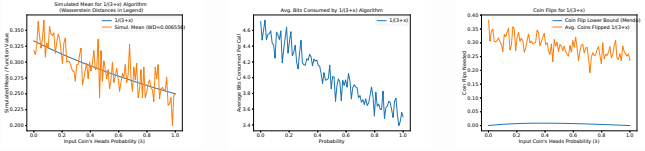
$$1/(1+x) \text{ (Even Parity)}$$



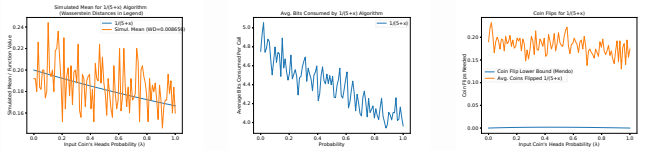
1/(1+x) (Two-Coin Special Case)



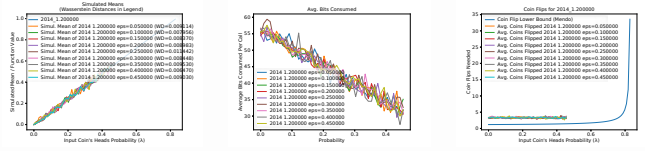
1/(3+x)



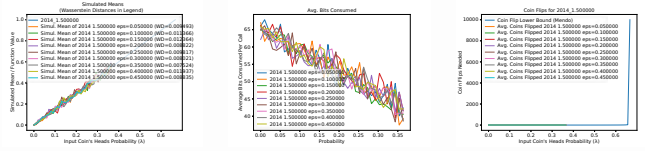
1/(5+x)



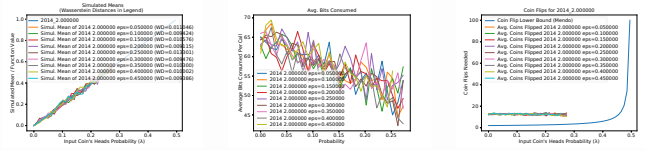
2014 1.200000
eps=0.050000



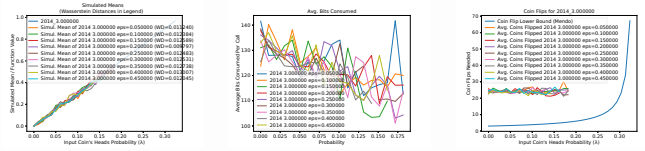
2014 1.500000
eps=0.050000



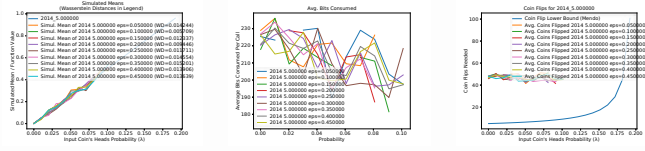
2014 2.000000
eps=0.050000



2014 3.000000
eps=0.050000



2014 5.000000
eps=0.050000



2014 Add. x+0.1

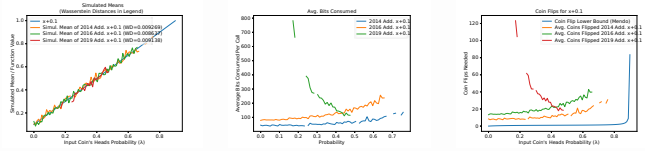


Figure 1 consists of three plots. The left plot, titled 'Dispersed Values (Observations minus Regression)', shows True Values (mm) on the y-axis (0 to 1.5) versus Predicted Values (mm) on the x-axis (0.5 to 1.5). It includes data for 2014-2016 (green dots) and 2017-2019 (red dots), with a blue line representing the regression. The middle plot, titled 'Avg. RMSE Computed', shows Average RMSE (mm) on the y-axis (0 to 10) versus Normalized Error on the x-axis (0.5 to 1.5). It includes data for 2014-2016 (green line) and 2017-2019 (red line), with a blue line representing the regression. The right plot, titled 'Avg. RMSE Computed', shows RMSE (mm) on the y-axis (0 to 80) versus Normalized Error on the x-axis (0.5 to 1.5). It includes data for 2014-2016 (green line) and 2017-2019 (red line), with a blue line representing the regression.

Figure 1 consists of three subplots comparing the proposed method (CUP Filter) with state-of-the-art methods (CUP Filter Lower Bound, CUP Filter Upper Bound, and CUP Filter) across different datasets and AUC values.

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

This plot shows the stratified mean AUC (std) for different methods across various datasets. The x-axis represents the AUC value, ranging from 0.6 to 0.8. The y-axis represents the stratified mean AUC (std), ranging from 0.6 to 0.8. The legend indicates the following methods and datasets:

- $\alpha=0.5$ (Blue line)
- 2014 data set of 2018 data $\alpha=0.5$ (Orange line)
- 2014 data set of 2018 data $\alpha=0.5$ (Green line)
- 2014 data set of 2018 data $\alpha=0.5$ (Red line)
- 2014 data set of 2018 data $\alpha=0.5$ (Blue line)

Top Right Plot: Average ROC Curves (std) vs AUC

This plot shows the average ROC curves (std) for different methods across various datasets. The x-axis represents the AUC value, ranging from 0.6 to 0.8. The y-axis represents the average ROC curves (std), ranging from 0.6 to 0.8. The legend indicates the following methods and datasets:

- 2014 data set of 2018 data $\alpha=0.5$ (Orange line)
- 2014 data set of 2018 data $\alpha=0.5$ (Green line)
- 2014 data set of 2018 data $\alpha=0.5$ (Red line)
- 2014 data set of 2018 data $\alpha=0.5$ (Blue line)

Bottom Plot: CUP Filter for AUC=0.5 vs CUP Filter for AUC=0.5

This plot shows the CUP Filter for AUC=0.5 for different methods across various datasets. The x-axis represents the CUP Filter for AUC=0.5, ranging from 0.6 to 0.8. The y-axis represents the CUP Filter for AUC=0.5, ranging from 0.6 to 0.8. The legend indicates the following methods and datasets:

- CUP Filter Lower Bound (Orange line)
- CUP Filter Upper Bound (Green line)
- CUP Filter (Red line)
- CUP Filter (Blue line)

[illegible][illegible]

Figure 1 consists of three subplots labeled (a), (b), and (c).
 Subplot (a) is a scatter plot titled "Scatter Plot of Predicted vs. Actual Mean (Covariance)". The x-axis is "Input C₁ Mean Probability (%)" ranging from 0.0 to 0.5. The y-axis is "Predicted Mean (Covariance)" ranging from 0.0 to 1.0. Data points are colored circles representing different input mean probabilities: 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5. A red diagonal line represents the ideal prediction. The points closely follow this line.
 Subplot (b) is a line plot titled "Avg. Mean Squared Error (MSE) vs. Probability". The x-axis is "Probability" ranging from 0.0 to 0.5. The y-axis is "Average Mean Squared Error (MSE)" ranging from 0.0 to 0.05. Multiple lines represent different input mean probabilities: 0.0 (blue), 0.1 (orange), 0.2 (green), 0.3 (red), 0.4 (purple), and 0.5 (brown). The MSE generally increases with probability, with the 0.5 input mean probability showing the highest error.
 Subplot (c) is a line plot titled "Cost Ratio vs. Probability". The x-axis is "Input C₁ Mean Probability (%)" ranging from 0.0 to 0.5. The y-axis is "Cost Ratio" ranging from 0 to 100. Multiple lines represent different input mean probabilities: 0.0 (blue), 0.1 (orange), 0.2 (green), 0.3 (red), 0.4 (purple), and 0.5 (brown). The cost ratio is highest at low probabilities and decreases as probability increases, with the 0.5 input mean probability showing the lowest cost ratio.

[illegible]

Figure 1 consists of three subplots comparing the proposed method (solid lines) with the baseline method (dashed lines) for various input cases. The top-left plot shows the 'Distorted Net Output (kPa)' versus 'Input Case's mean Probability (%)' for cases 1 through 6. The top-right plot shows the 'Avg. Case's Net Output (kPa)' versus 'Probability' for the same cases. The bottom plot shows the 'Case Net Output (kPa)' versus 'Input Case's mean Probability (%)' for the same cases. In all plots, the proposed method generally follows the baseline method closely, with some deviations at higher probabilities.

Figure 1 consists of two plots, (a) and (b), comparing the proposed model with existing models.

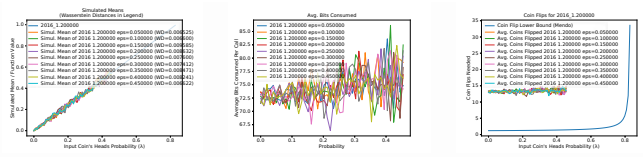
Plot (a) shows the Standard Error of the Mean (SEM) on the y-axis (ranging from 0.00 to 0.02) versus Input Concentration (mg/L) on the x-axis (ranging from 0.00 to 0.60). The legend includes:

- Prop. (red line)
- Linear (blue line)
- Linear + 1st Order (green line)
- Linear + 2nd Order (orange line)
- Linear + 3rd Order (purple line)
- Linear + 4th Order (brown line)
- Linear + 5th Order (pink line)
- Linear + 6th Order (grey line)
- Linear + 7th Order (light blue line)
- Linear + 8th Order (light green line)
- Linear + 9th Order (light orange line)
- Linear + 10th Order (light purple line)
- Linear + 11th Order (light brown line)
- Linear + 12th Order (light pink line)
- Linear + 13th Order (light grey line)
- Linear + 14th Order (light blue line)
- Linear + 15th Order (light green line)
- Linear + 16th Order (light orange line)
- Linear + 17th Order (light purple line)
- Linear + 18th Order (light brown line)
- Linear + 19th Order (light pink line)
- Linear + 20th Order (light grey line)

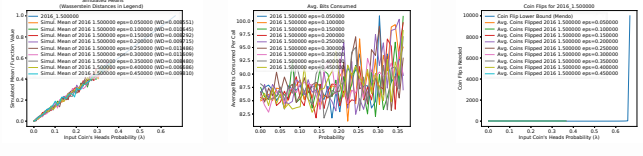
Plot (b) shows the Average Bio-Concentration Factor (BCF) on the y-axis (ranging from 0 to 200) versus Input Concentration (mg/L) on the x-axis (ranging from 0.00 to 0.60). The legend includes:

- Prop. (red line)
- Linear (blue line)
- Linear + 1st Order (green line)
- Linear + 2nd Order (orange line)
- Linear + 3rd Order (purple line)
- Linear + 4th Order (brown line)
- Linear + 5th Order (pink line)
- Linear + 6th Order (grey line)
- Linear + 7th Order (light blue line)
- Linear + 8th Order (light green line)
- Linear + 9th Order (light orange line)
- Linear + 10th Order (light purple line)
- Linear + 11th Order (light brown line)
- Linear + 12th Order (light pink line)
- Linear + 13th Order (light grey line)
- Linear + 14th Order (light blue line)
- Linear + 15th Order (light green line)
- Linear + 16th Order (light orange line)
- Linear + 17th Order (light purple line)
- Linear + 18th Order (light brown line)
- Linear + 19th Order (light pink line)
- Linear + 20th Order (light grey line)

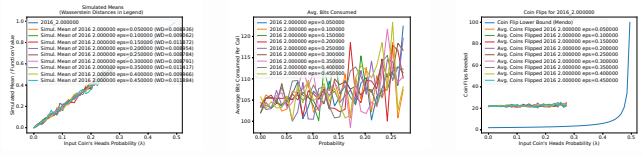
```
2016 1.200000
eps=0.050000
```



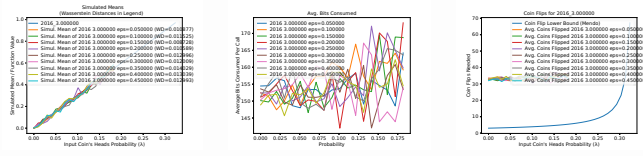
```
2016 1.500000
eps=0.050000
```



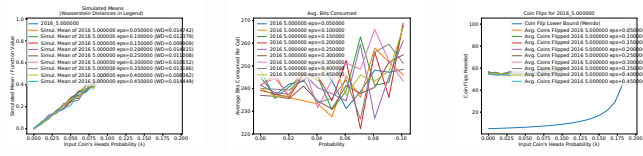
```
2016 2.000000
eps=0.050000
```



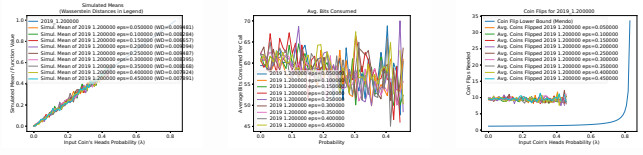
```
2016 3.000000
eps=0.050000
```



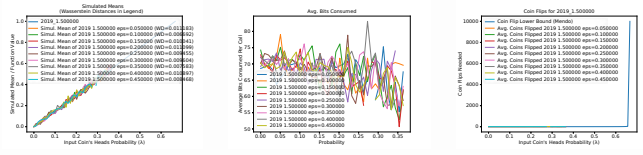
```
2016 5.000000
eps=0.050000
```



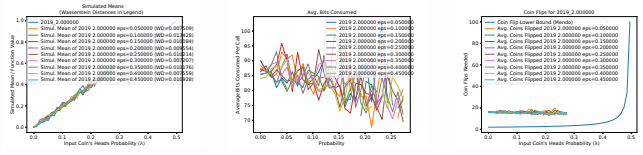
```
2019 1.200000
eps=0.050000
```



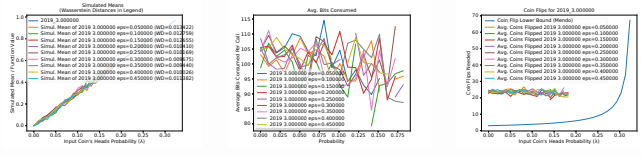
```
2019 1.500000
eps=0.050000
```



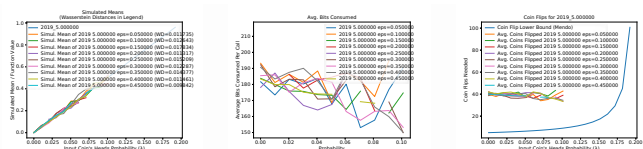
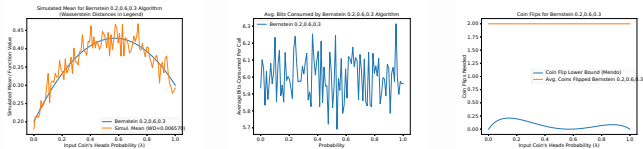
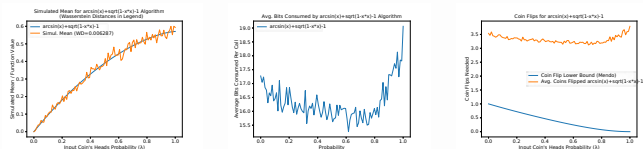
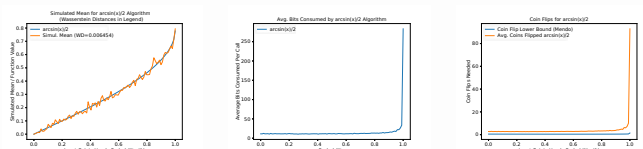
```
2019 2.000000
eps=0.050000
```



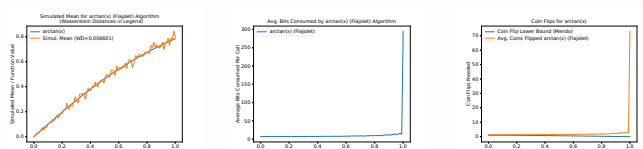
2019 3.000000
eps=0.050000



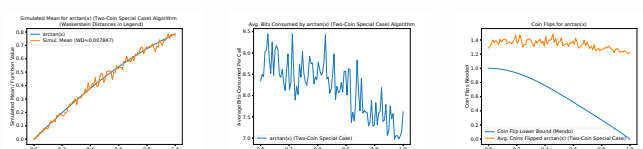
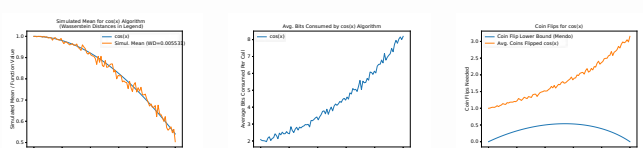
```
2019 5.000000
eps=0.050000
```

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

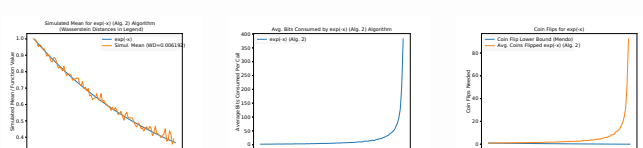
arctan(x)
(Flajolet)



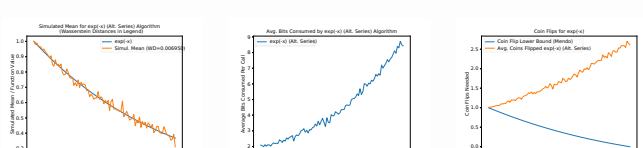
arctan(x) (Two-Coin Special Case)

 $\cos(x)$ 

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



exp(-x) (Alt.
Series)



Estimated Mean for each of (Proposed) Algorithm

Standard Error (times) Risk

Logarithmic Coefficient (β)

Exact Mean (0.001-0.005)

Prop. Mean (0.001-0.005)

Average Coefficient (β)

Logarithmic Coefficient (β)

CPU Time (sec)

Logarithmic Coefficient (β)

Exact Mean (0.001-0.005)

Prop. Mean (0.001-0.005)

Figure 10 consists of three subplots:

- Left Plot:** Scalability of the proposed algorithm. The x-axis is 'Input Data Size (Number of Nodes)' ranging from 0.0 to 1.0. The y-axis is 'Scalability of the Proposed Algorithm' ranging from 0.0 to 1.0. The plot shows two curves: 'Proposed' (blue line with circles) and 'Baseline' (orange line with circles). Both curves start at 1.0 and decrease as the input data size increases, with the proposed algorithm maintaining higher scalability than the baseline.
- Middle Plot:** Avg. Bits Consumed by the proposed algorithm. The x-axis is 'Input Data Size' ranging from 0.0 to 1.0. The y-axis is 'Average Bits Consumed' ranging from 0.0 to 1.0. The plot shows two curves: 'Proposed' (blue line with circles) and 'Baseline' (orange line with circles). Both curves start at 0.0 and increase as the input data size increases, with the proposed algorithm consuming fewer bits than the baseline.
- Right Plot:** Gap Error for input data size. The x-axis is 'Input Data Size (Number of Nodes)' ranging from 0.0 to 1.0. The y-axis is 'Gap Error' ranging from 0.0 to 1.0. The plot shows two curves: 'Proposed' (blue line with circles) and 'Baseline' (orange line with circles). Both curves start at 0.0 and increase as the input data size increases, with the proposed algorithm showing a lower gap error than the baseline.

Serialized Mean for $\log_2(x+1)$ (Pigeon) Algorithm
Serialized Mean for $\log_2(x+1)$ (Proposed) Algorithm

Serialized Mean for $\log_2(x+1)$ (Pigeon) Algorithm
Serialized Mean for $\log_2(x+1)$ (Proposed) Algorithm

Cost Ratio for $\log_2(x+1)$

Cost-Flip Greedy
Avg. Cost-Flip Greedy

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' from 0.0 to 1.0. The y-axis is 'Disabled flow capacity (kA)' from 0.0 to 1.0. It shows two curves: 'points 1-10' (blue line with dots) and 'points 100-1000' (orange line with dots). Both curves start at (0,0) and increase monotonically, 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' from 0.0 to 1.0. The y-axis is 'Average flow capacity (kA)' from 0.0 to 1000.0. It shows two curves: 'points 1-10' (blue line with dots) and 'points 100-1000' (orange line with dots). Both curves start at approximately 1000 kA and drop sharply to near zero by iteration 0.2, remaining there until iteration 1.0.
 Subplot (c) is titled 'Error: Error (kA) vs. Iteration number'. The x-axis is 'Iteration number' from 0.0 to 1.0. The y-axis is 'Error (kA)' from 0 to 400. It shows two curves: 'points 1-10' (blue line with dots) and 'points 100-1000' (orange line with dots). Both curves start at approximately 400 kA and drop sharply to near zero by iteration 0.2, remaining there until iteration 1.0.

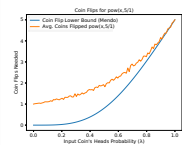
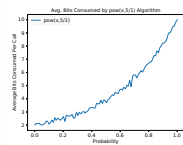
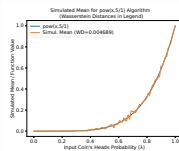
[illegible]

Figure 1 consists of four subplots showing the performance of the proposed S-VI algorithm compared to a baseline (p-sims).

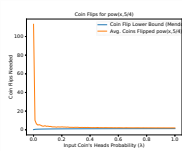
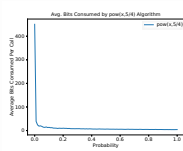
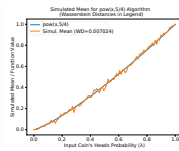
- Top-left plot:** Stratified Mean for p-sims, S-VI Algorithm. The y-axis is 'Stratified Mean for p-sims, S-VI Algorithm' (0.0 to 1.0) and the x-axis is 'Input Coin's Heads Probability (x)' (0.0 to 1.0). The legend indicates 'p-sims' (blue line) and 'S-VI' (orange line). The S-VI line is consistently above the p-sims line, indicating better performance.
- Top-right plot:** Avg. Bits Consumed by p-sims, S-VI Algorithm. The y-axis is 'Avg. Bits Consumed by p-sims, S-VI Algorithm' (0.0 to 1.0) and the x-axis is 'Probability (x)' (0.0 to 1.0). The legend indicates 'p-sims' (blue line) and 'S-VI' (orange line). The S-VI line is consistently below the p-sims line, indicating lower bit consumption.
- Bottom-left plot:** Mean RLE for p-sims, S-VI Algorithm. The y-axis is 'Mean RLE for p-sims, S-VI Algorithm' (0.0 to 1.0) and the x-axis is 'Probability (x)' (0.0 to 1.0). The legend indicates 'p-sims' (blue line) and 'S-VI' (orange line). The S-VI line is consistently below the p-sims line, indicating lower RLE.
- Bottom-right plot:** Cost Flips for p-sims, S-VI. The y-axis is 'Cost Flips for p-sims, S-VI' (0.0 to 1.0) and the x-axis is 'Input Coin's Heads Probability (x)' (0.0 to 1.0). The legend indicates 'p-sims' (blue line) and 'S-VI' (orange line). The S-VI line is consistently below the p-sims line, indicating lower cost.

Figure 1 consists of three subplots. Subplot (a) is a line graph titled 'Simulated Mean for penta-K5 Algorithm' showing 'Distorted Mean (fraction of bits)' on the y-axis (0.0 to 1.0) versus 'Input Bits Probability (bits)' on the x-axis (0.0 to 1.0). It compares 'Distorted Mean (penta-K5)' (blue line with circles) and 'Simul. Mean (MD=0.0000001)' (orange line with circles). Both lines are nearly identical, starting at (0,0) and ending at (1,1). Subplot (b) is a line graph titled 'Avg. Bits Consumed by penta-K5 Algorithm' showing 'Average Bits Consumed (bits)' on the y-axis (0 to 10) versus 'Input Bits Probability (bits)' on the x-axis (0.0 to 1.0). The blue line with circles starts at approximately 10 bits for 0.0 probability and decreases to about 1 bit for 1.0 probability. Subplot (c) is a line graph titled 'Avg. Bits Consumed by penta-K5 Algorithm' showing 'Average Bits Consumed (bits)' on the y-axis (0 to 4.5) versus 'Input Bits Probability (bits)' on the x-axis (0.0 to 1.0). It compares 'Dist. Bits Consumed (penta-K5)' (blue line with circles) and 'Avg. Bits Consumed (penta-K5)' (orange line with circles). Both lines start at approximately 4.5 bits for 0.0 probability and decrease to about 1 bit for 1.0 probability, with the blue line being slightly higher than the orange line.

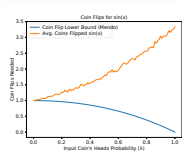
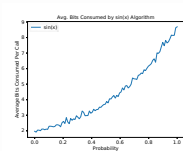
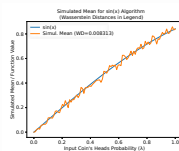
pow(x,5/1)



pow(x,5/4)



sin(x)



sqrt(x)

