HISTORY-BASED PIECEWISE APPROXIMATION SCHEME FOR PROCEDURES

Aurangzeb and Rudolf Eigenmann

Outline

- Motivation for approximating procedures
- Contributions of the paper
  - History-based piecewise scheme for approximating procedures for speed
    - 2 flavors – uniform and non-uniform
    - Four realizations with features
  - Results of approximating 90 functions from GSL (GNU Scientific Library)
  - Results on benchmarks and real applications

Motivation

- Approximate Computing
- Procedures with pure functional behavior
  - Mathematical and Scientific functions
  - Other functions
- State of the art
  - Numerical Analysis Techniques

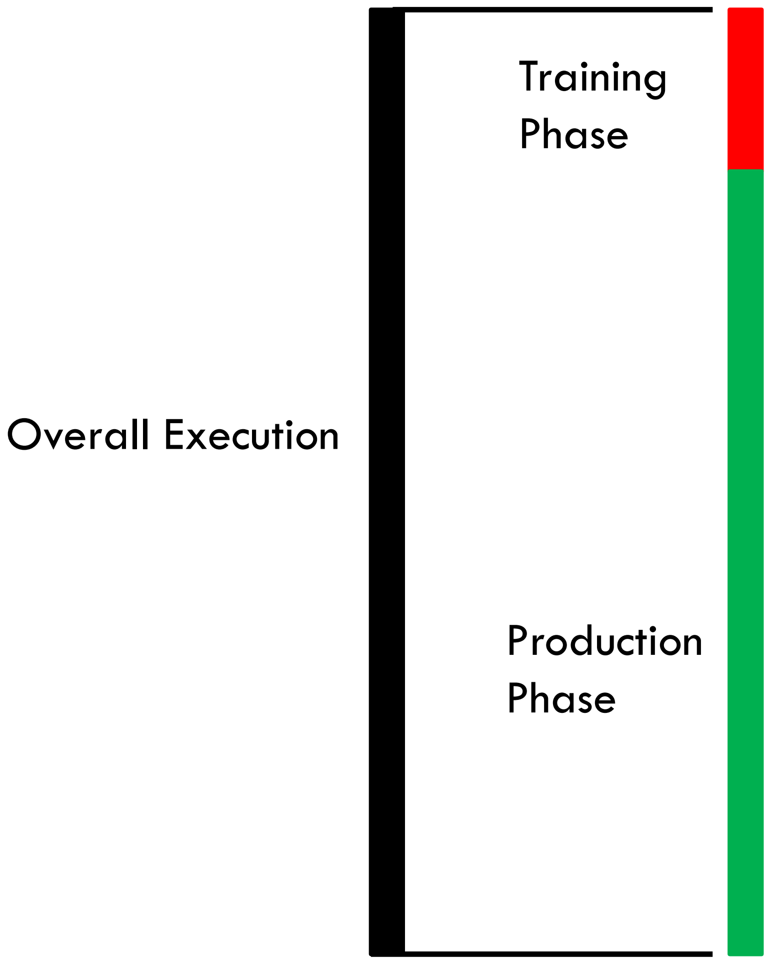
Contributions

- History-based piecewise scheme for approximating procedures for speed
  - 2 flavors – uniform and non-uniform
  - Four realizations with features
- Results of approximating 90 functions from GSL (GNU Scientific Library)
- Results on benchmarks and real applications

History-based Piecewise Approximation

- Approximating function with one polynomial does not generally give good approximation
- Piecewise schemes give better results
- Forming regions and choosing polynomials is difficult
- History-based piecewise approximation scheme
  - Forms regions based on history
  - Uses low-order polynomials for approximation
- Types
  - Non-uniform piecewise approximation
  - Uniform piecewise approximation

History-based Piecewise Approximation

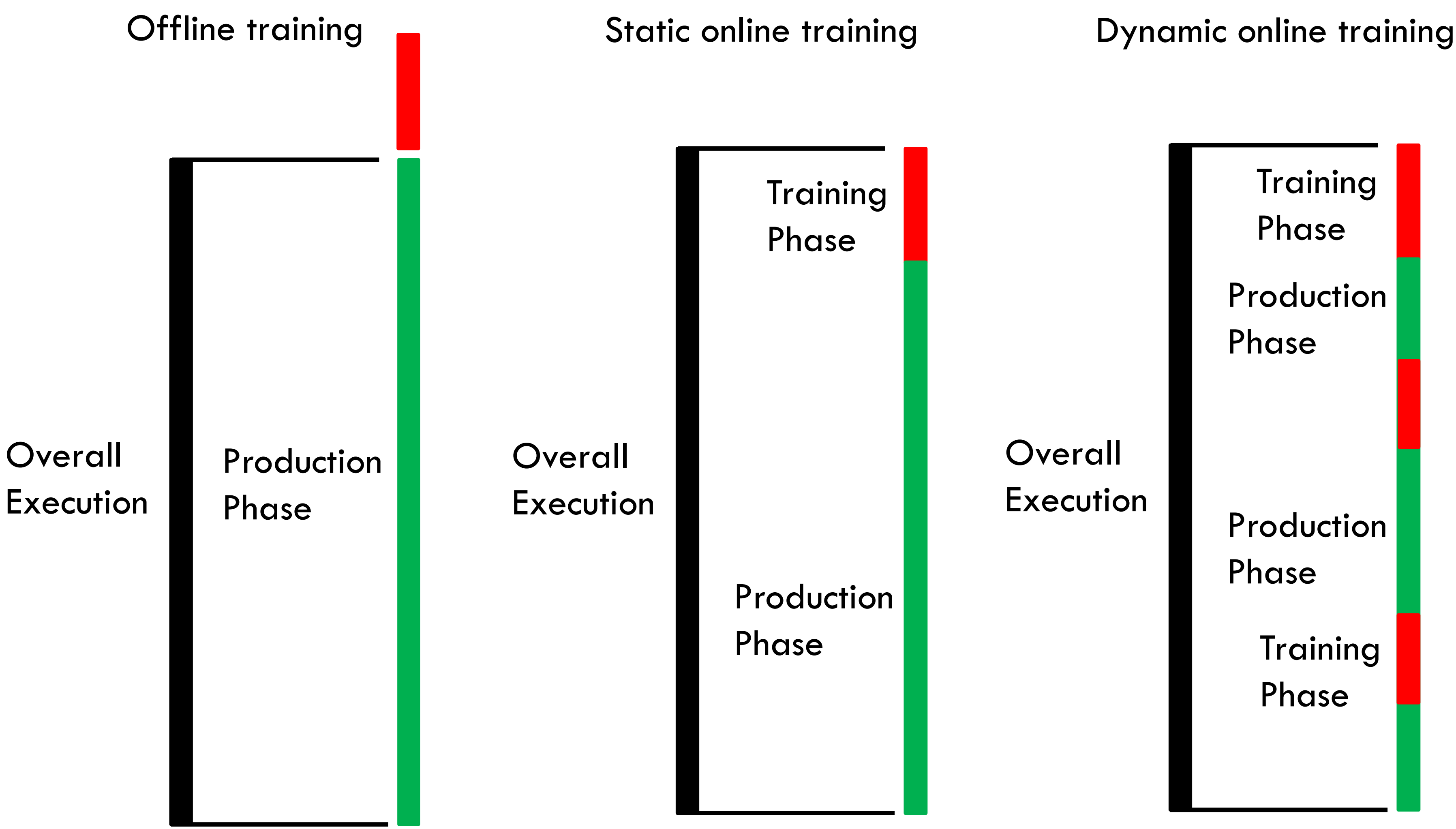


Collect history to form regions

Search the history to find the region an input falls into

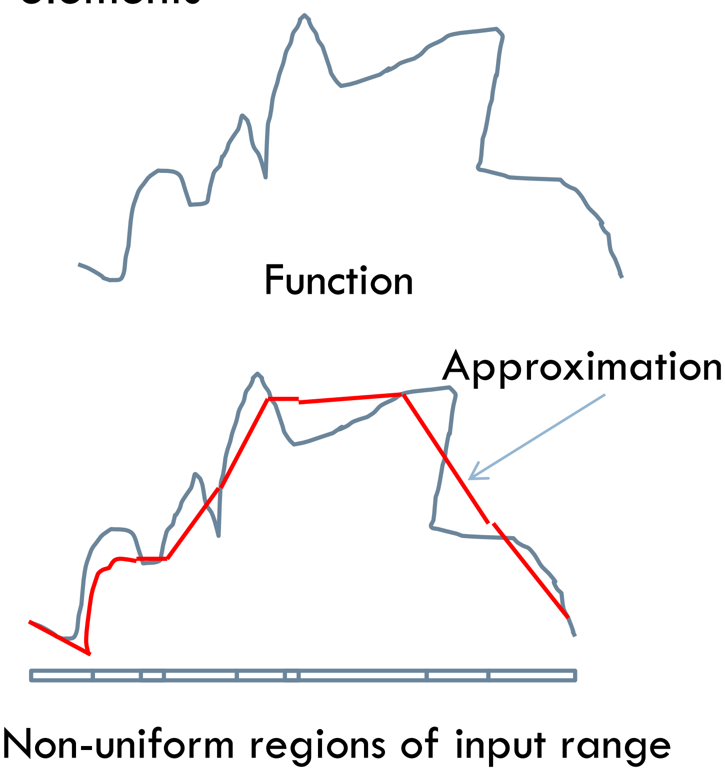
Approximate within that region using low-order polynomials

Different Training Scenarios



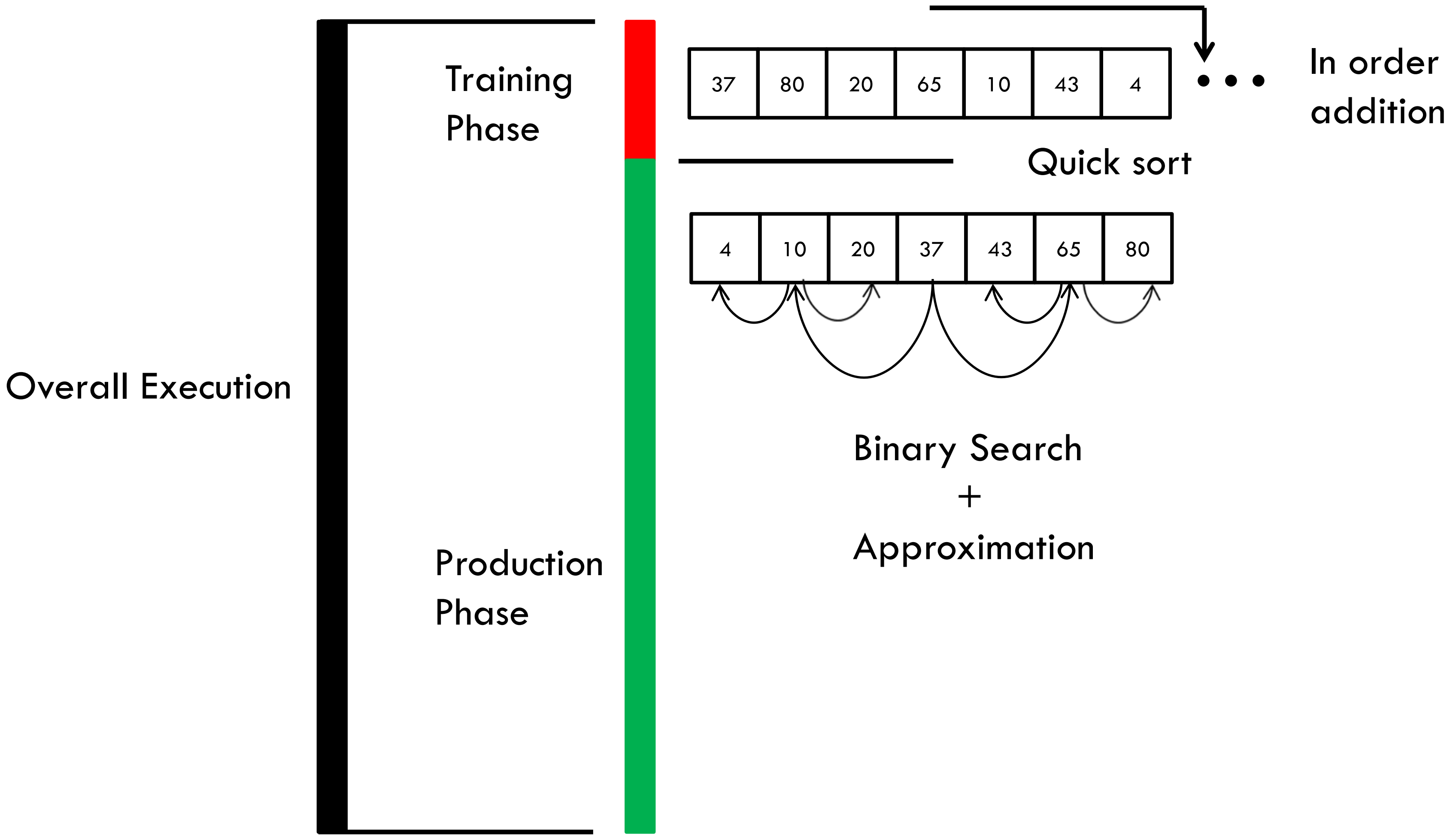
Non-uniform Piecewise Approximation

- Non-uniform regions based on **learnt history** of input-output behavior
- Allows for non-uniform concentration of history elements
- Approximation scheme
  - Constant
  - 1-degree polynomial (interpolation)
  - higher degree polynomials are possible
- Realizations
  1. Binary search on sorted array
  2. Binary search tree (BST)
  3. Red-black tree (RBT)



Non-uniform Piecewise Approximation (3/5)

- Binary search on sorted array



Non-uniform Piecewise Approximation (4/5)

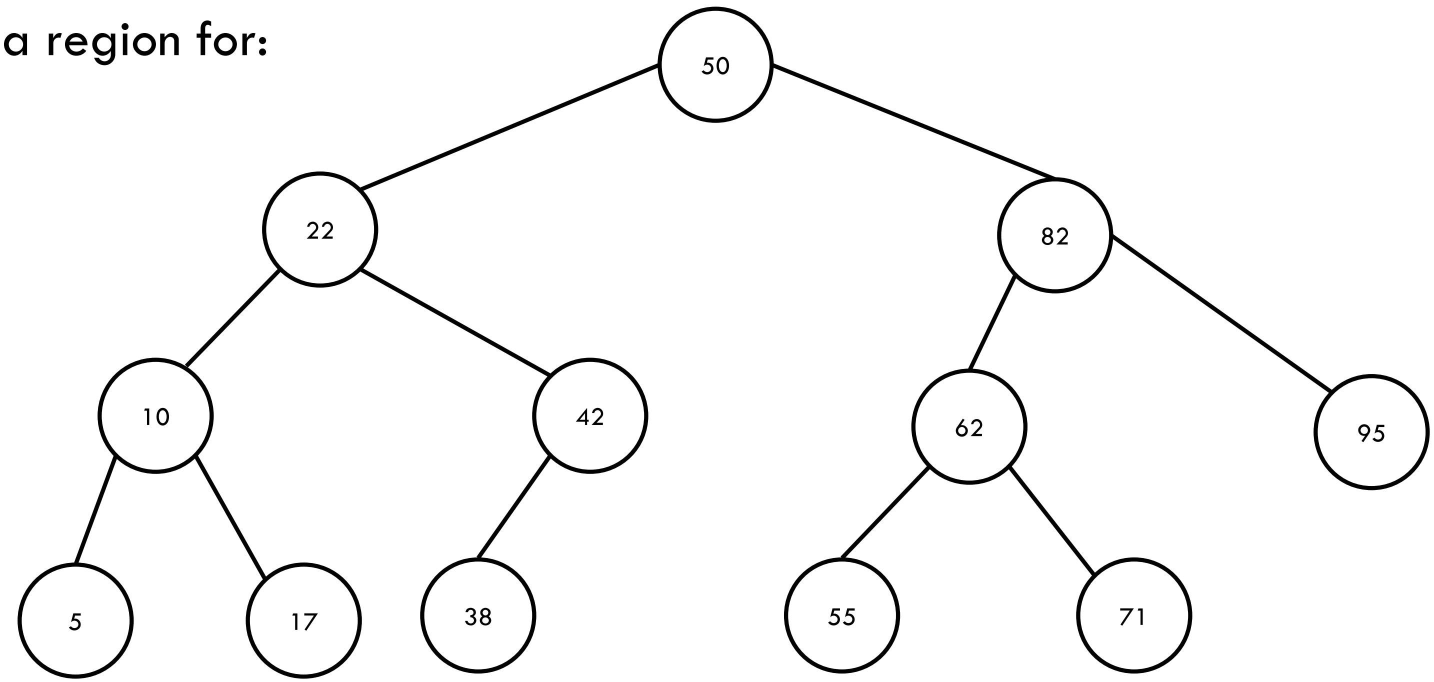
- Binary Search Tree (BST)

- Finding a region for:

  40

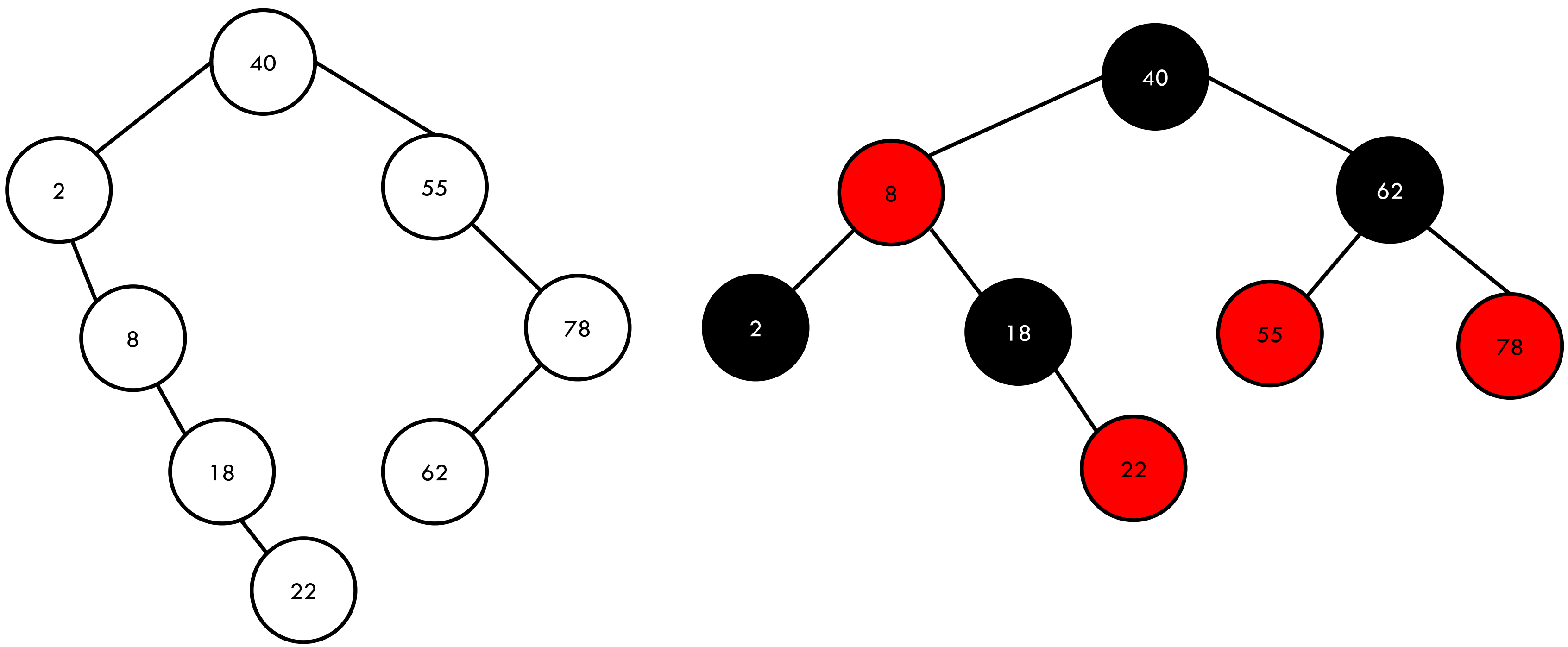
  18

  53



Non-uniform Piecewise Approximation (5/5)

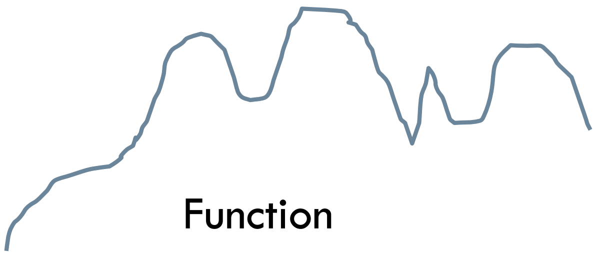
- Red-Black Tree (RBT)
  - BST can become a list in the worst case
  - RBT is approximately balanced

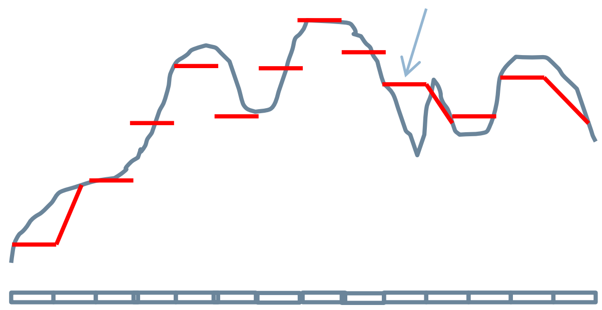


Sequence: 40, 2, 55, 8, 18, 22, 78, 62

Uniform Piecewise Approximation (1/2)

- Uniform regions
- Currently formed based on profiling information, possible to guess from first **m** function invocations
- Does not allows for non-uniform concentration of history elements
- Approximation scheme
  - 0-degree polynomial (currently one value per region is stored)
  - higher degree polynomials are possible





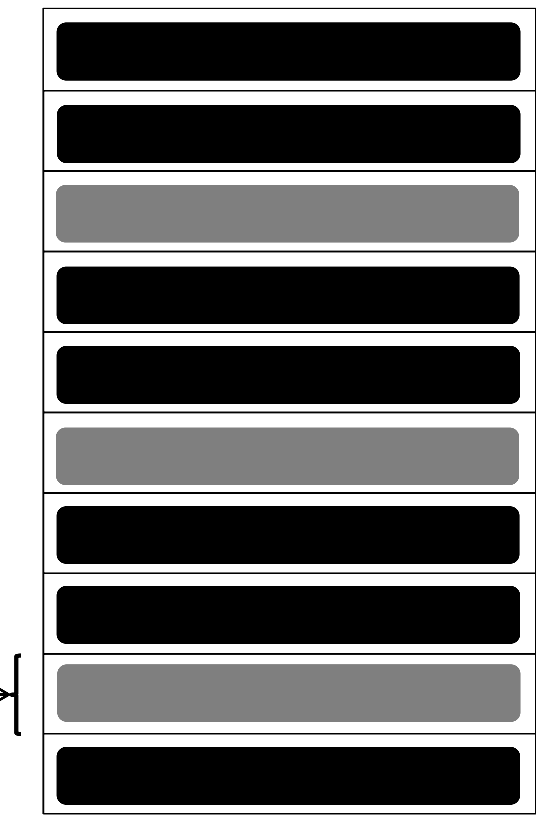
Uniform Piecewise Approximation (2/2)

- Realization via a hash table

Black – untrained

Gray – trained

Region size



Feature Comparison of Schemes

Scheme	Storage & lookup	Fixed/Extendable	Lookup Complexity	Supports non-uniform concentration?	Order
Non-uniform	Array + Binary search	Fixed	O(log N)	Yes	Sorted
Non-uniform	Binary search tree (BST)	Extendable	> O(log N)	Yes	Binary search tree property
Non-uniform	Red-black tree	Extendable	~O(log N)	Yes	BST + red-black property
Uniform	Hash-table	Fixed	O(1)	No	Ranges are sorted

Evaluation (1/6)

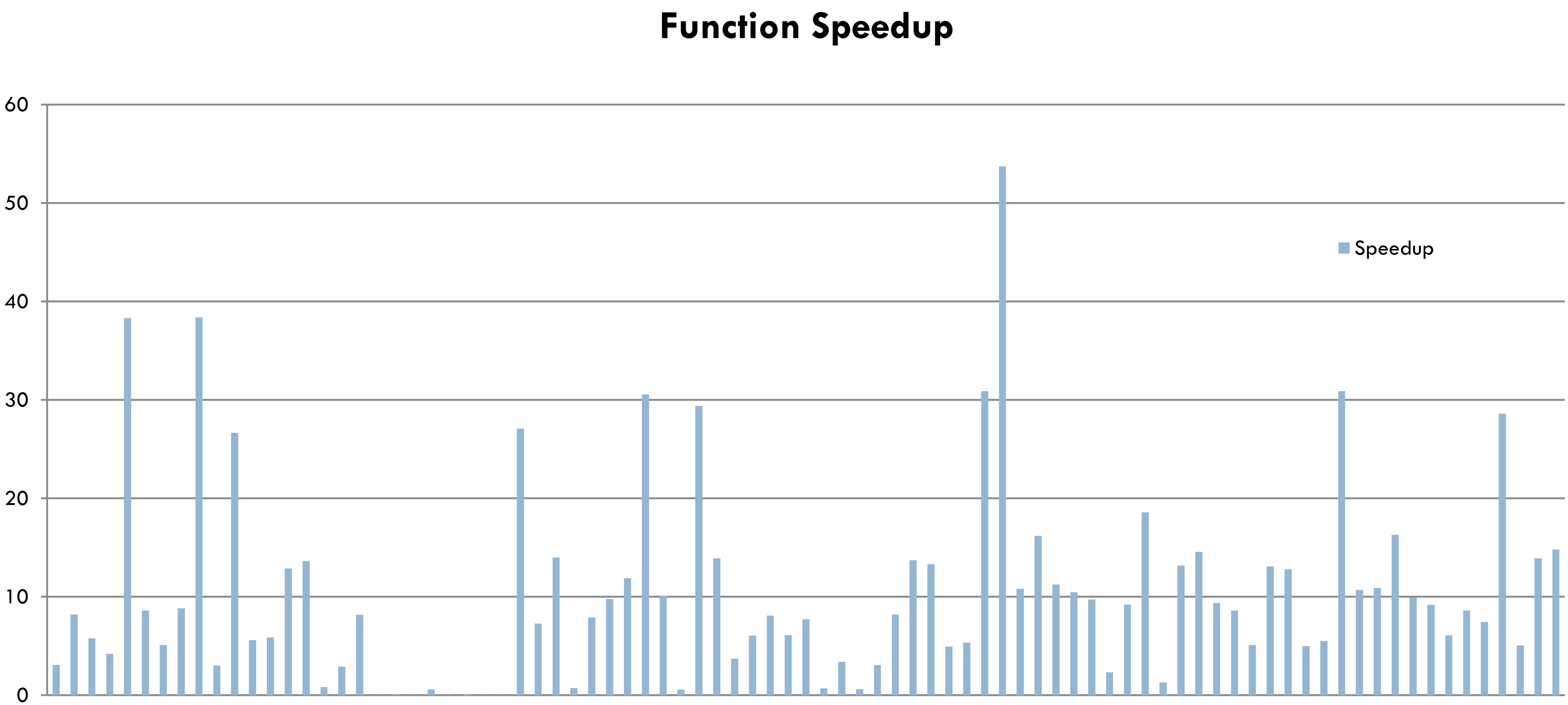
- System
  - Intel Core2 Duo CPU running at 3 GHz, 6144 KB cache, 4 GB RAM
  - Ubuntu 12.04
- Testing
  - Functions
    - Mathematical and scientific functions from GSL (GNU Scientific Library)
  - Benchmarks
    - Blackscholes and Swaptions from PARSEC benchmark suite
  - Applications
    - CNN-HDD: Neural network application for handwritten digit detection

Evaluation (2/6)

- Functions from GSL
  - Tested 90 functions
  - Selected a realistic input range for each function
  - Called each function 1,000,000 times with random inputs
  - Results for uniform approximation scheme
    - Able to speed up 92% functions
    - For 71% functions, avg speedup is 9.3x for an avg RMS error of 0.06
    - For another 15%, avg speedup is 9.5x for an avg RMS error of 0.49
    - The rest report large (188) RMS error for an average speedup of 12x
    - Functions with exponential behavior have higher absolute RMS

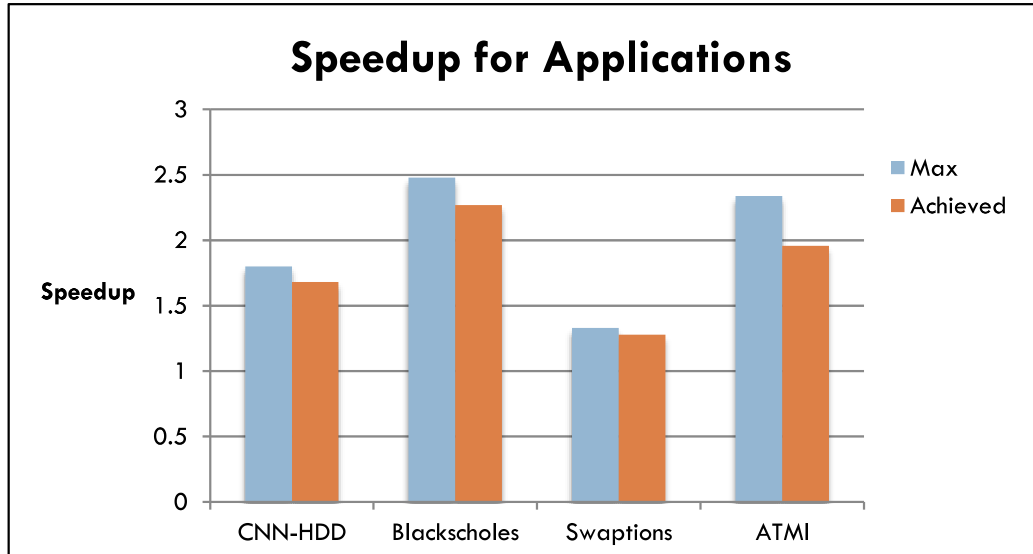
Evaluation (3/6)

- Functions from GSL



Evaluation (4/6)

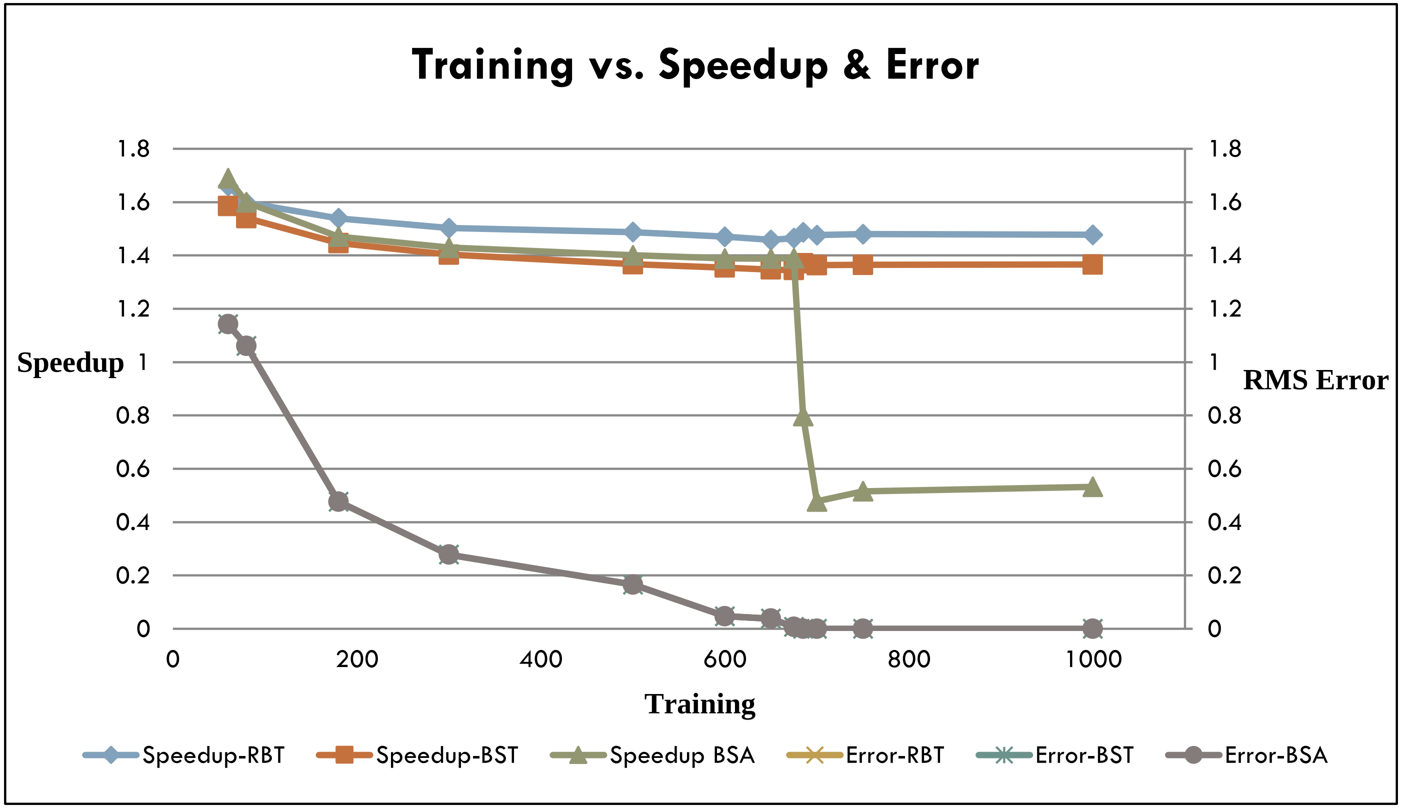
- Results for applications
- Avg speedup = 1.74x
- Avg %error = 0.5%



Application	Domain	Function	#invocations	Max Possible speedup	Error	Error Metric	Application speed up	# regions, region size, and memory (KB)
CNN-HDD	Machine Learning	tanh	8,010,000	1.8	0.02%	%undetected images	1.68	702, 0.05, 5.6
Blackscholes	Financial	CNDF	13,107,200	2.48	0.048	RMS Error	2.27	7200, 0.005, 57.6
Swaptions	Financial	CumNormalInv	76,800,000	1.33	0.005	RMS Error	1.28	200, 0.005, 1.6

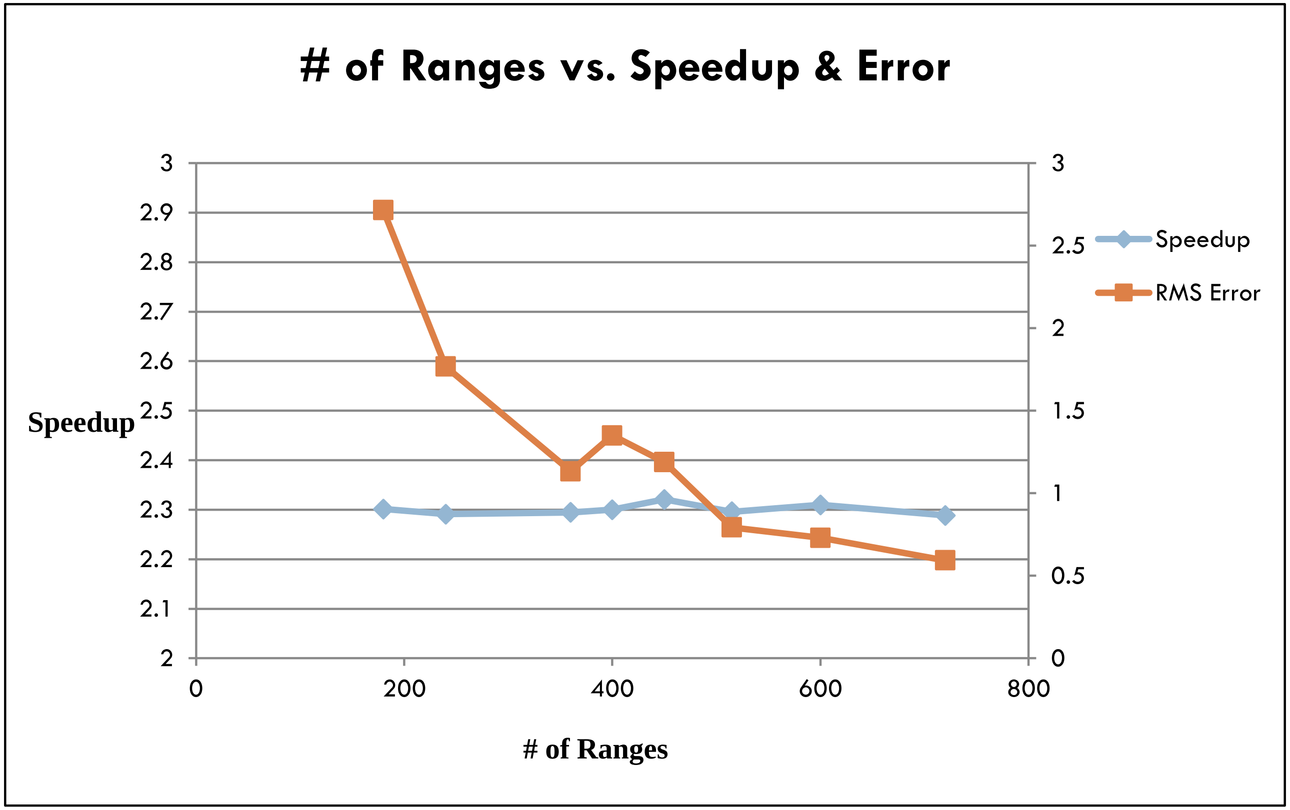
Evaluation (5/6)

- Comparison of non-uniform techniques using Blackscholes application
- Effect of training on speedup and Error



Evaluation (6/6)

- Effect of number of regions on speedup and error in the uniform scheme



Uniform scheme for Blackscholes application

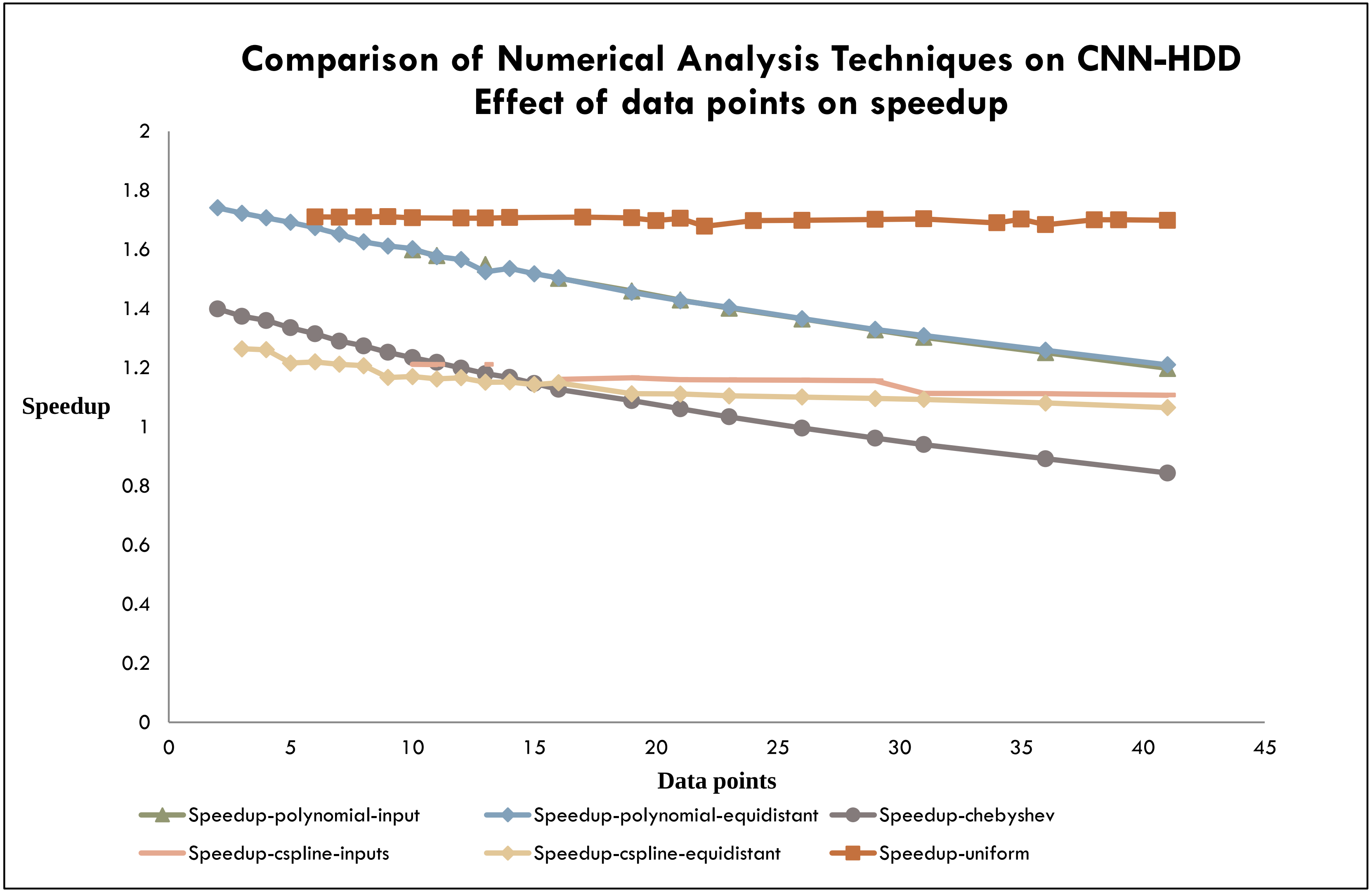
Related Work (1/6)

- Numerical Analysis
  - Polynomial Approximation (monomials (1, x, x^2, x^3, …), Chebyshev, others)
  - Splines (cubic splines, others)
- Machine Learning
  - ANN (Artificial Neural Networks), SVMs (Support Vector Machines), Fitness approximation in Genetic Algorithms
- Approximate Computing
  - Neural network based approach
  - Approximate memoization (Paraprox)

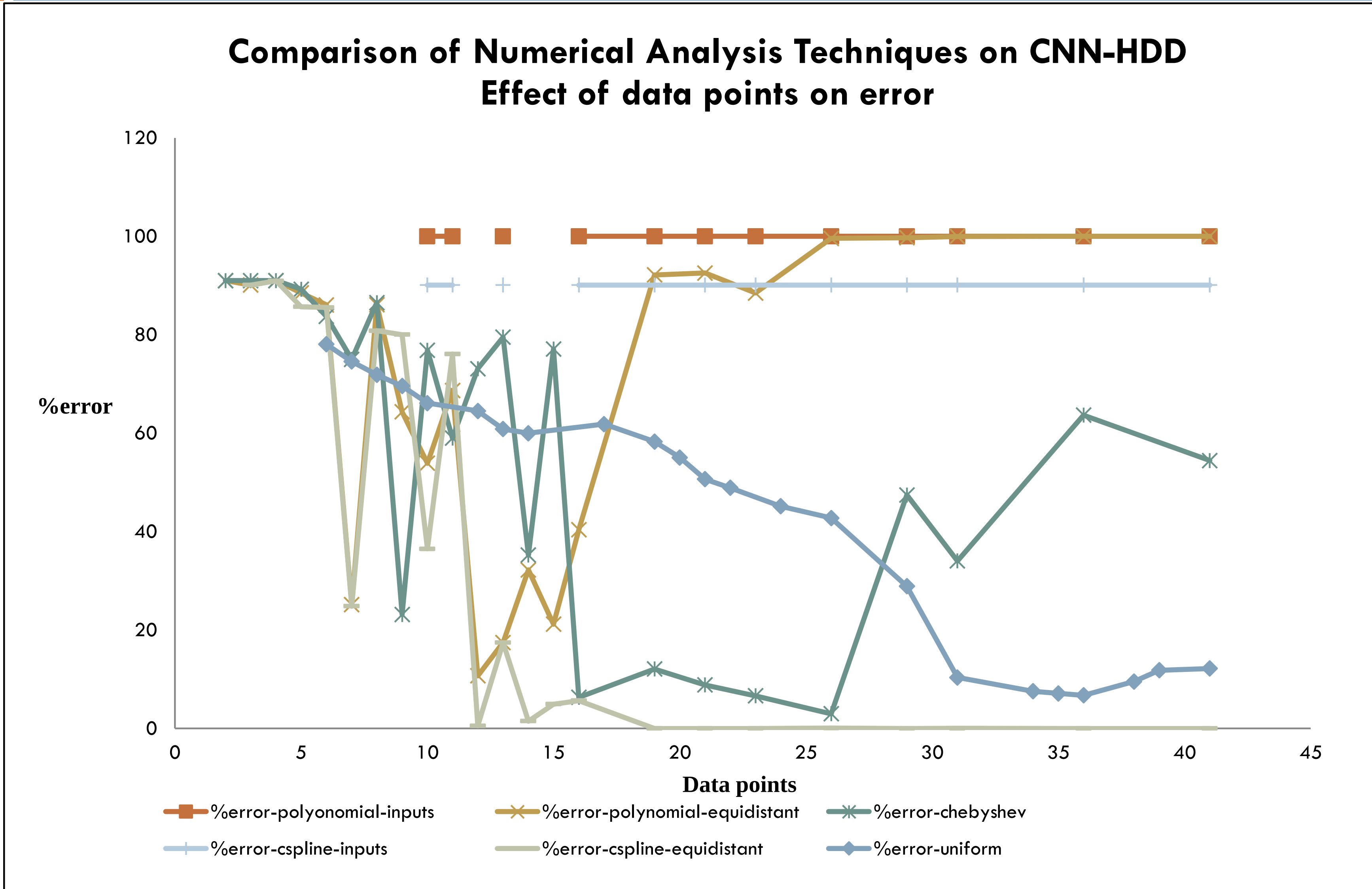
Related Work (2/6)

- Comparison with Numerical Analysis techniques
  - Polynomial approximation using monomial basis ($1, x, x^2, x^3, \ldots$)
    - First n+1 inputs
    - Evenly spaced data points over the entire input range
  - Chebyshev polynomial approximation
  - Cublic splines
    - First n+1 inputs
    - Evenly spaced intervals

Related Work (3/6)

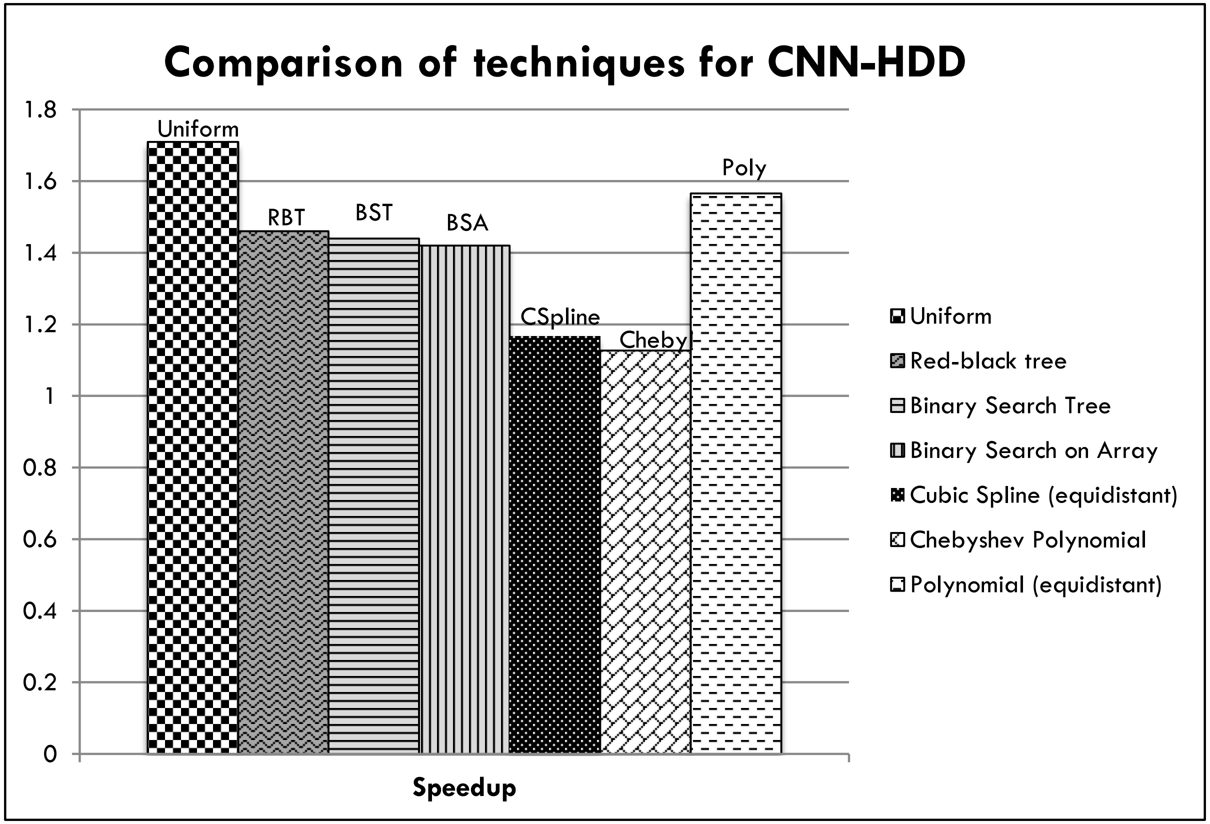


Related Work (4/6)



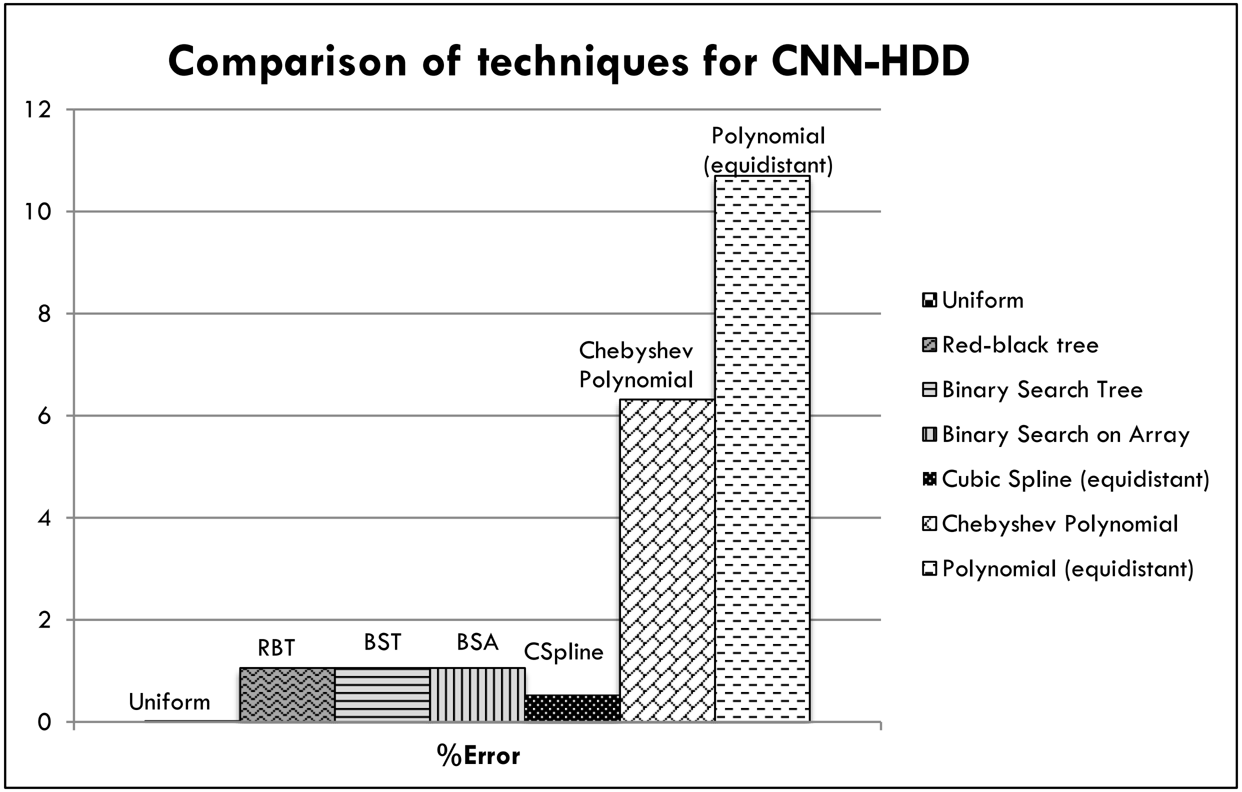
Related Work (5/6)

- Comparison with Numerical Analysis techniques on CNN-HDD application



Related Work (6/6)

- Comparison with Numerical Analysis techniques on CNN-HDD application



Future Work

- Support for 2nd degree polynomial
- Support for dynamic online training
- Automatic code generation and parameter selection

Thanks!