Analysis of Variable Selection Methods on Scientific Cluster Measurement Data

Jonathan Wang

University of California, Berkeley
&
Scientific Data Management Research Group
Computational Research Division
Lawrence Berkeley National Laboratory

Application: Palomar Transient Factory (PTF)

- Comprehensive transient detection system
  - Survey cameras
  - Realtime data reduction pipeline
  - Identify astronomical transients
- Observational data processed through NERSC
  - 38 checkpoints
  - Execution logs of runtime
- Effort to understand resource requirements
  - Model performance of data pipeline

PTF Performance Model

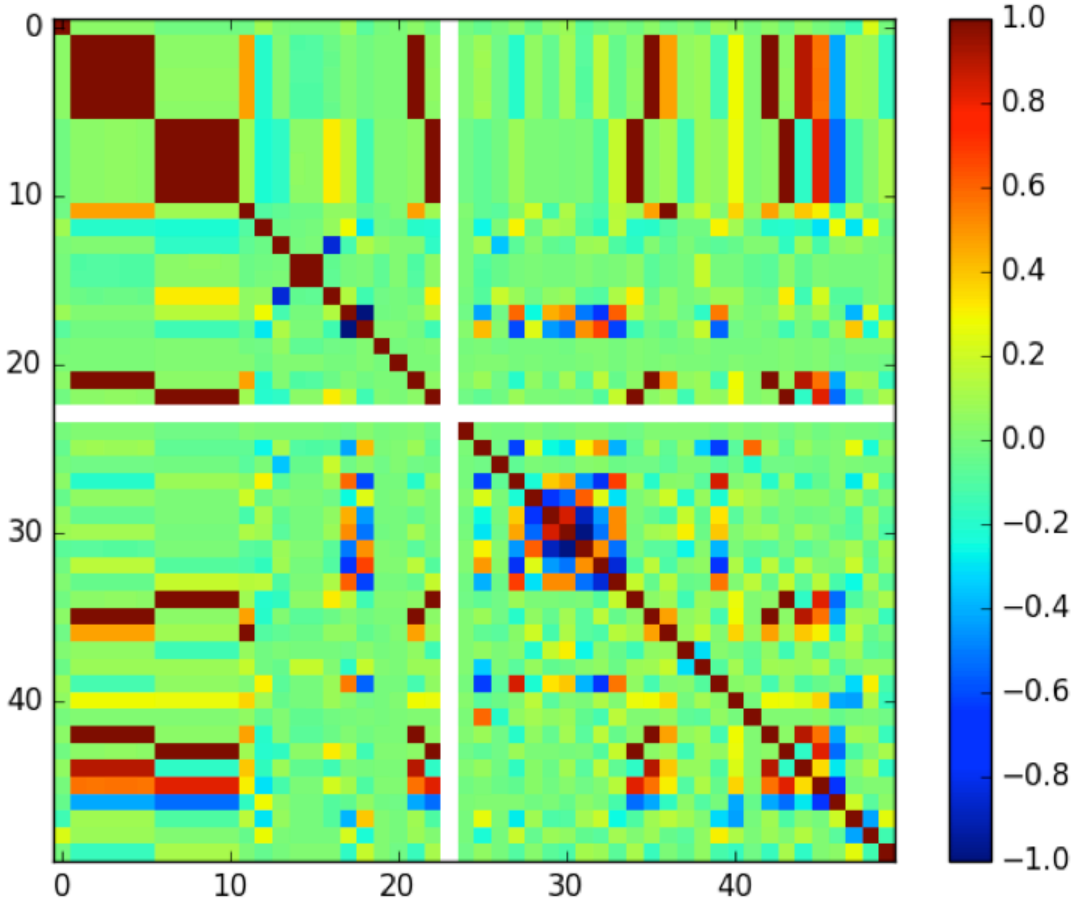
- Use PTF logs to predict execution time
  - Identify bottlenecks
  - Slowdown after checkpoint 15
- Goal: Predict execution time of checkpoints 16-38
- Input Features: Observational features and execution time for checkpoints 0-15

Motivation

- **Use variable selection methods to find optimal variable subset**
  - Reduce performance model training time (fewer features)
  - Improve prediction accuracy by removing noisy variables
- **Improve variable selection runtime**
  - Parallelize variable selection
  - Eliminate redundant variables quickly

Data Exploration

Feature correlation matrix

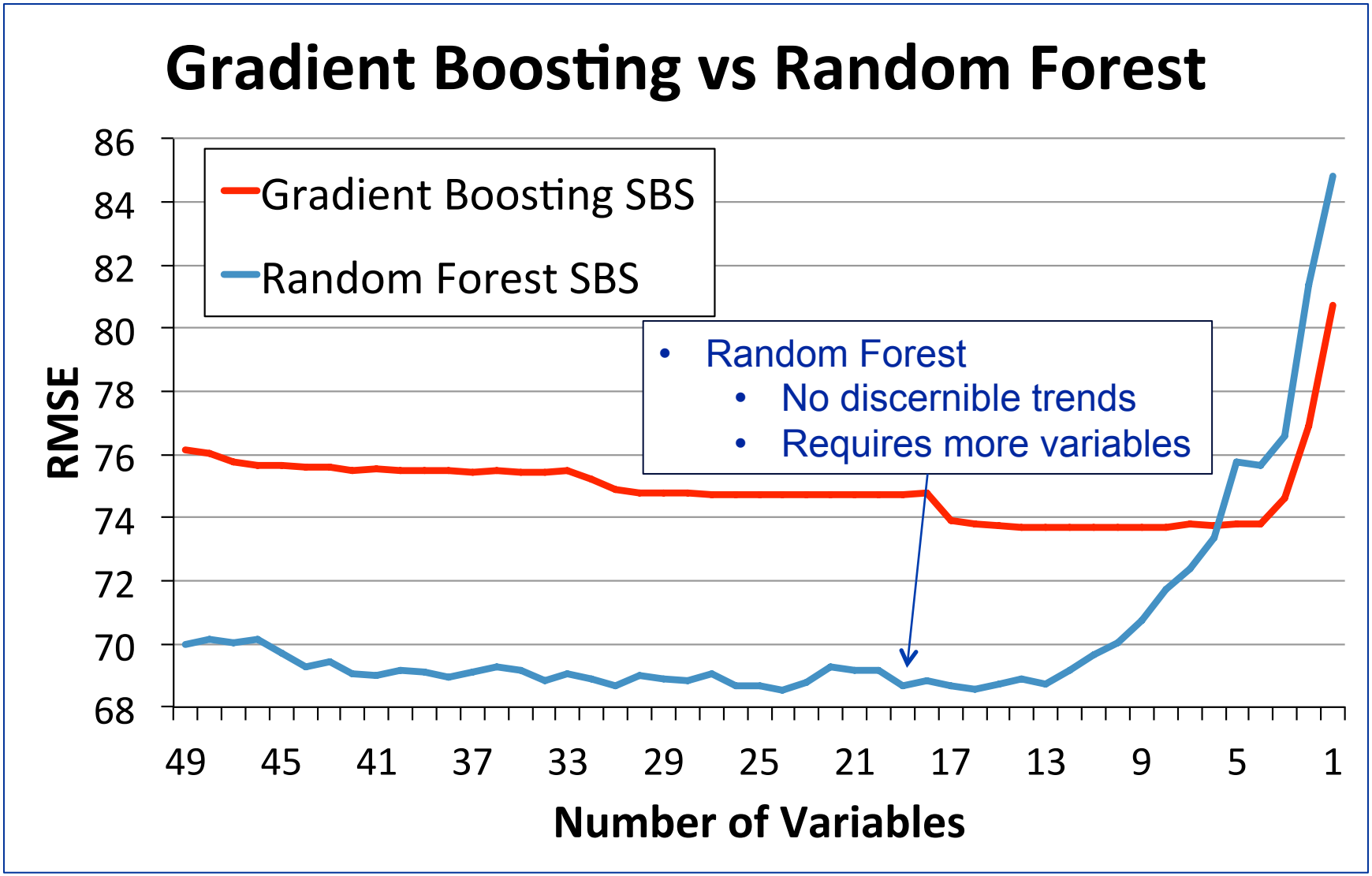


Multiple strongly correlated features (this is just a subset of features)

Prediction Model

- **Random Forest vs Gradient Boosting**
  - Random Forest has lower bias
  - 100x higher variance (0.2 v 0.002)
  - Inconsistent subset selection
- **Can offset bias by predicting with Random Forest after selecting subset**
- **Select prediction model that gives more information**
  - Consistent set of variables selected by importance
  - Less influence from model variance

Prediction Model



Variable Selection Methods

- **Methods from scikit-learn library**
  - RFE: Recursive Feature Elimination
  - Univariate: Build up model using F Regression
  - Importance: Based on Gini Impurity of Random Forest
- **Methods implemented in Spark**
  - Sequential Forward Selection (SFS): greedy selection from empty set
  - Sequential Backward Selection (SBS): greedy elimination from full set

***SFS and SBS were implemented in Spark to be parallelized**

Sequential Backward Selection (SBS)

- **Objective**
  - Remove single variable at each iteration
  - Select best size n-1 subset (lowest error)
- **Implementation**
  - Start with subset of all variables
  - For each variable train and test model without variable (parallelizable)
  - Select subset with lowest Root Mean Squared Error
  - Repeat until subset is of desired size

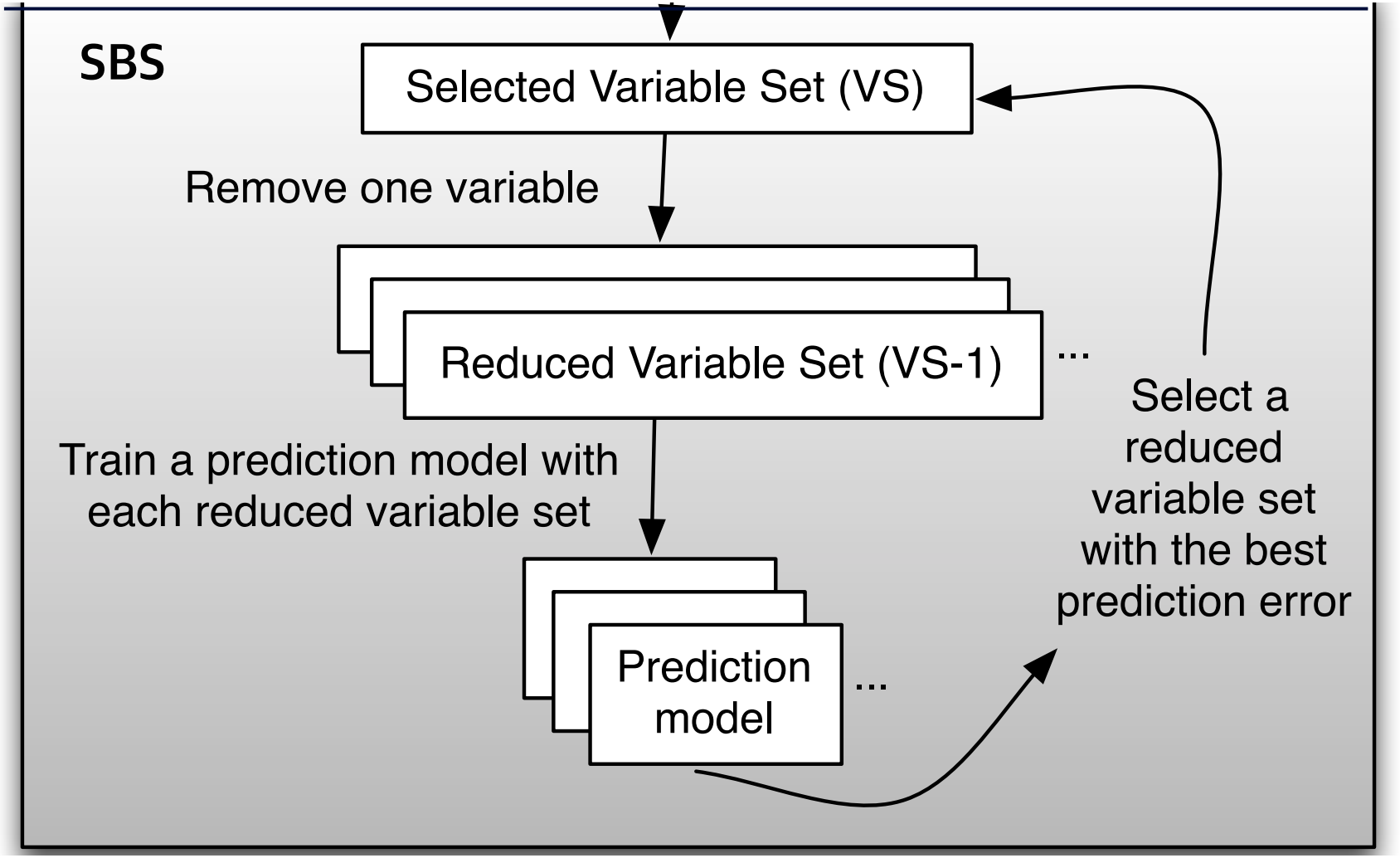
Experimental Setup

- Lawrencium HPC system at LBNL
  - 56 nodes
    - 2 12-core Intel Xeon E5-2670 CPUs
    - 64 GB memory
- Utilized 3 nodes for PTF experiments
  - Parallelize for one core per variable
  - Maximize parallelization on PTF data

SBS Parallelization

- **Parallelization**
  - Test each variable in subset at each iteration
  - One core for each variable to maximize parallelization
- **Improvement**
  - 18 hours (65020 sec) without parallelization
  - < 1 hour (2727 sec) when parallelized
  - 20x improvement (not 50x) since number of variables decreases after each iteration

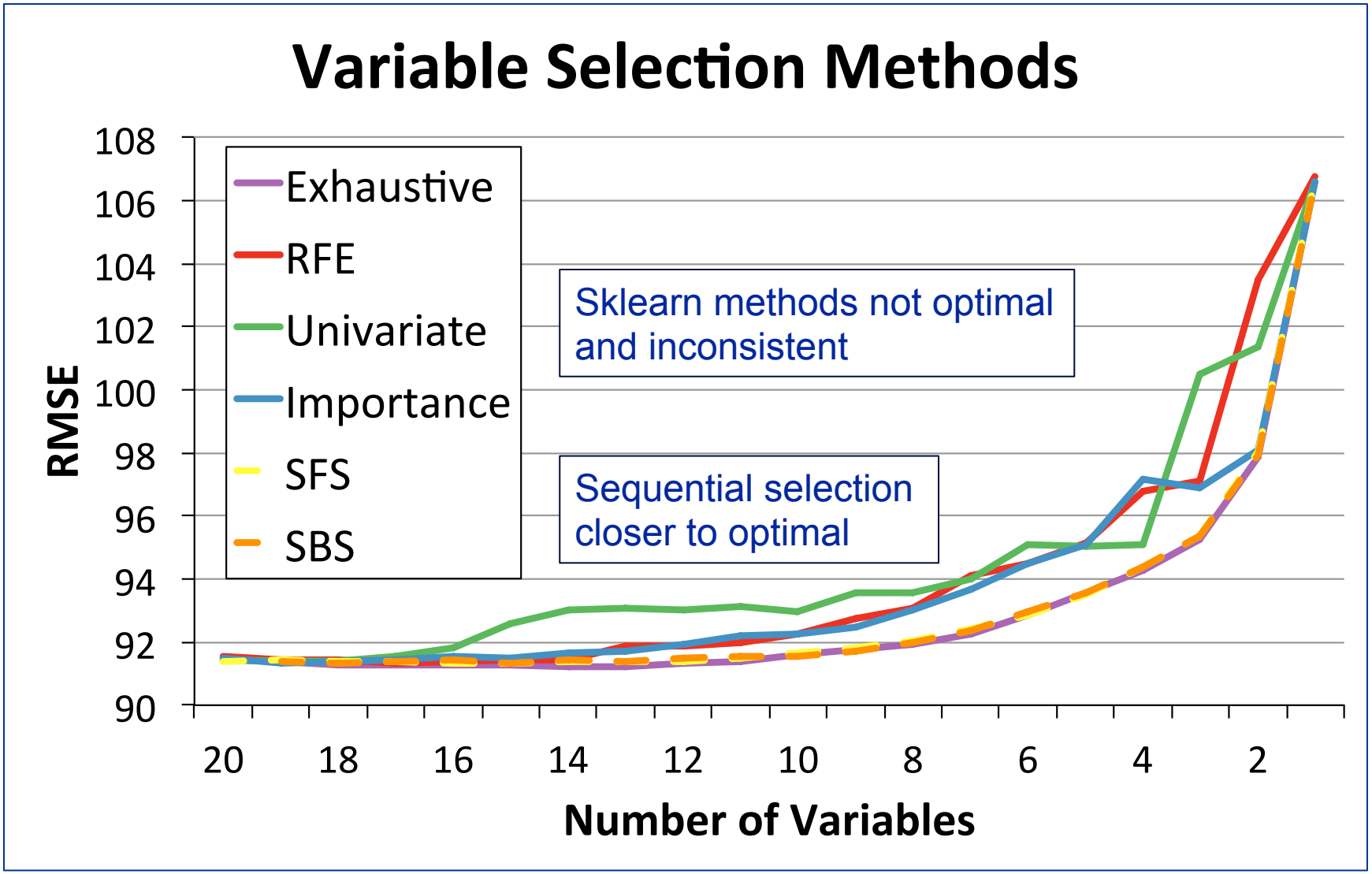
Sequential Backward Selection



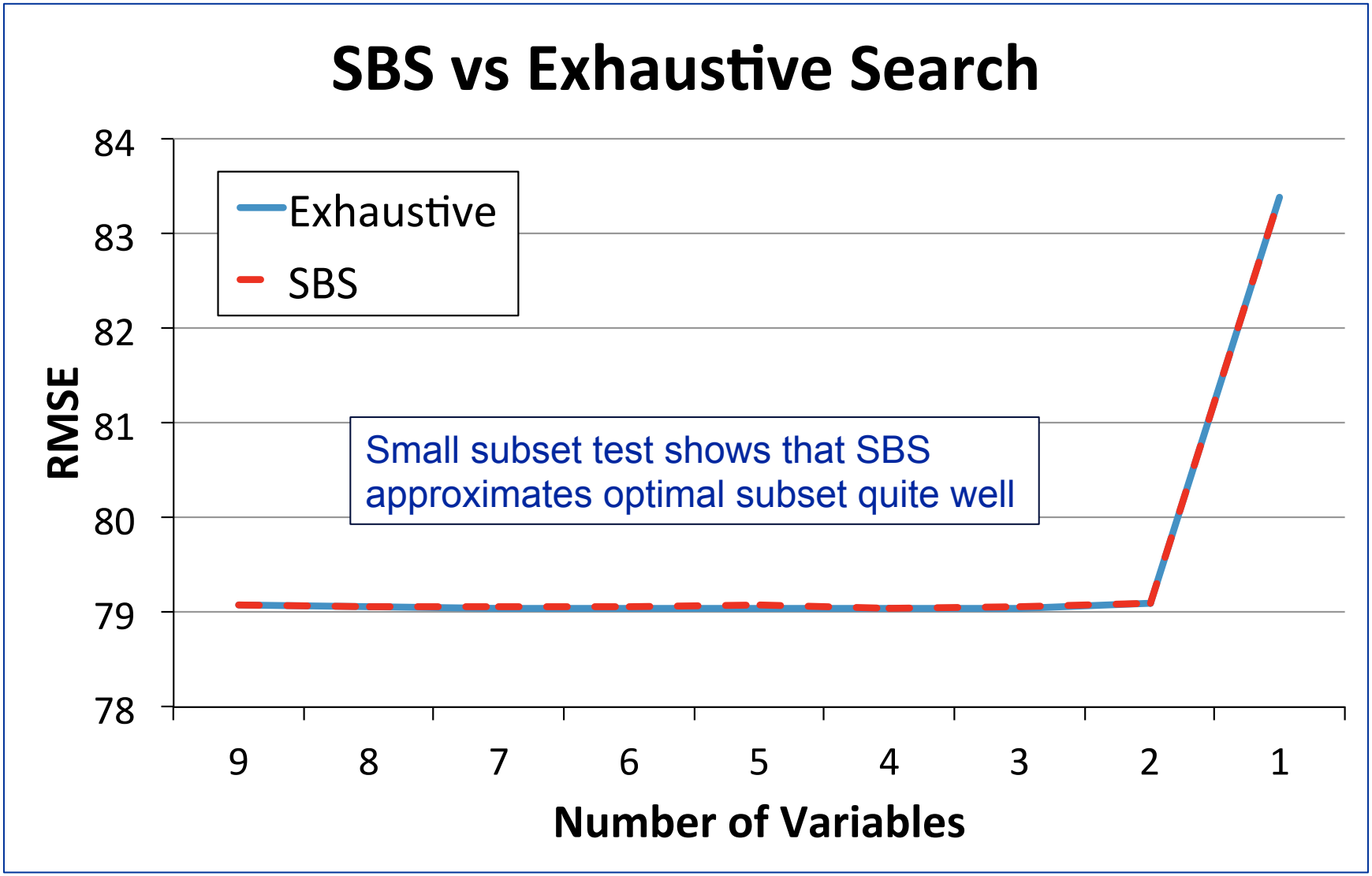
Variable Selection

- **Exhaustive Selection tests every combination of variables**
  - Determines optimal variable subset
  - Can only be completed on small variable sets due to exponential runtime
- **Ideal Selection Method**
  - Identifies optimal subset
  - Consistent improvement shows subset is not affected by variance

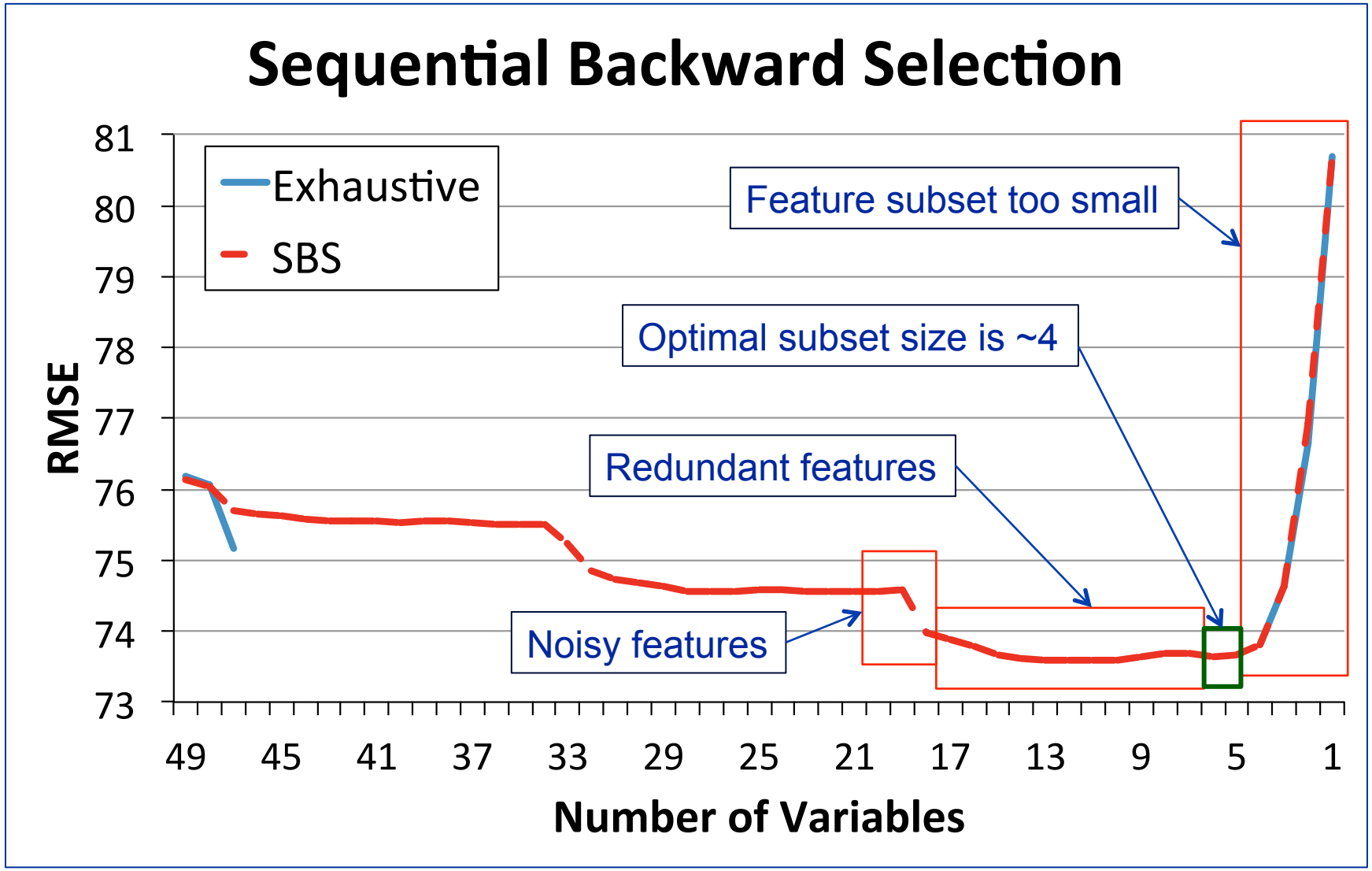
Variable Selection



SBS Performance



SBS Performance



Selected Variable Subsets

- **Significant features (feature index)**
  - ec_long (44) – ecliptic longitude
  - medsky (27) – median sky background
    - Brightness of sky affects visibility of transients
  - Time for checkpoints 0-15 (0)
  - gal_lat (47) – galactic latitude

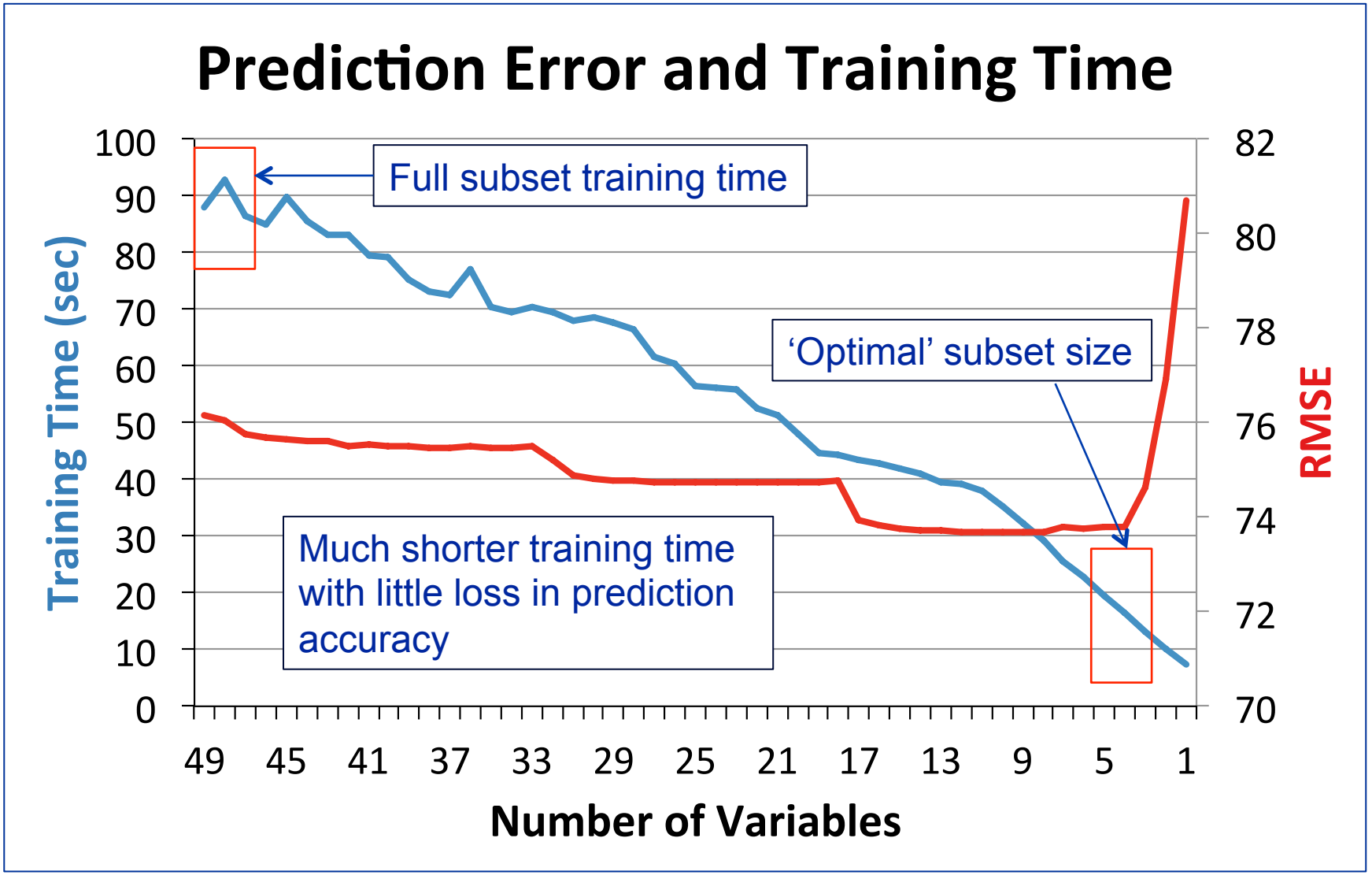
Selected Variable Subsets

- **Exhaustive**
  - {44} : RMSE = 80.706
  - {27, *47*} : RMSE = 76.637
  - {0, 27, 44} : RMSE = 74.622
  - {0, 27, 44, 47} : RMSE = 73.805
- **SBS**
  - {44} : RMSE = 80.706
  - {27, *44*} : RMSE = 76.932
  - {0, 27, 44} : RMSE = 74.622
  - {0, 27, 44, 47} : RMSE = 73.804

SBS Results

- Achieves comparable prediction error as exhaustive search
  - Small variable set test
  - Same final variable subsets
- Consistent trends of changing prediction error
  - Identify noisy and redundant variables
  - Decreasing prediction error until minimum subset size

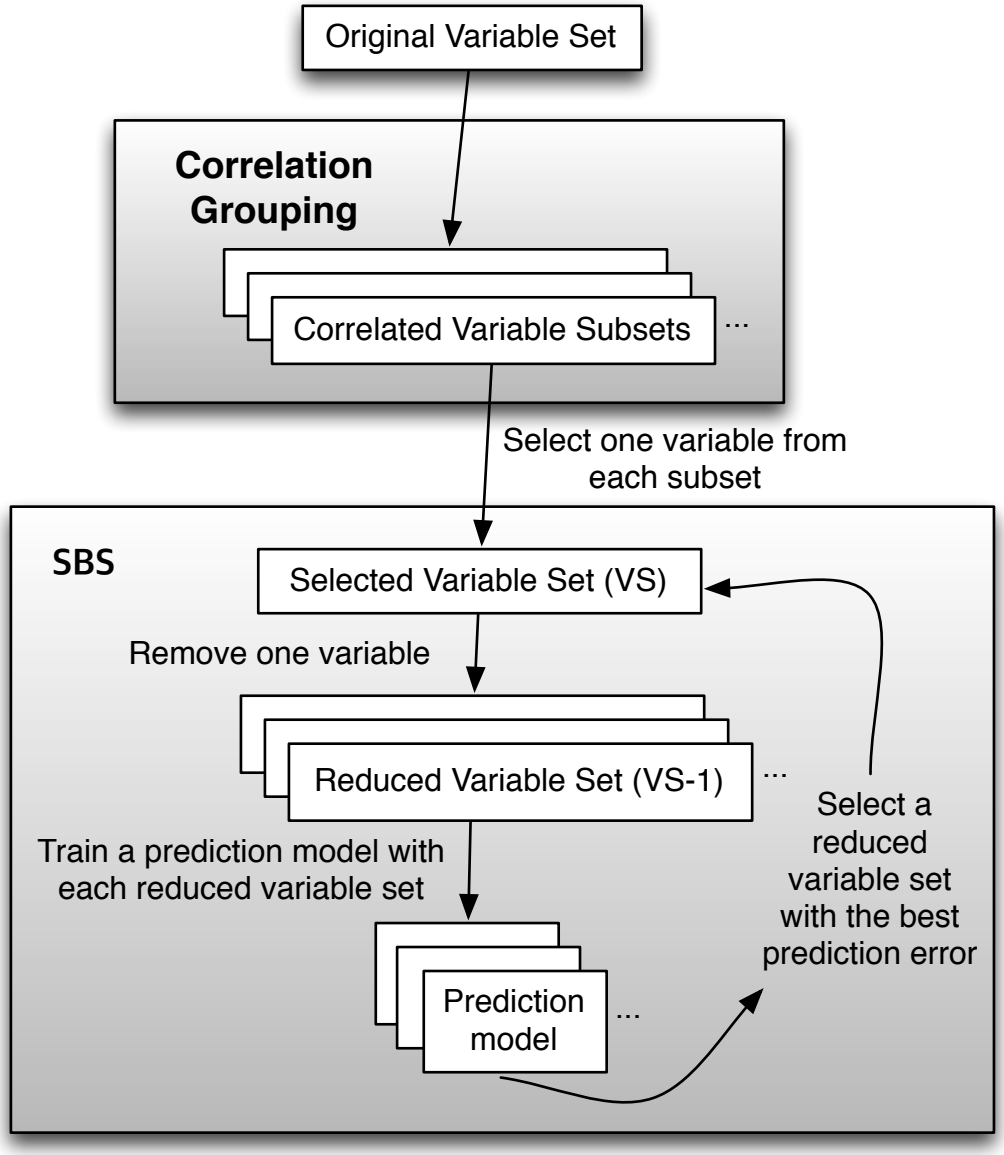
SBS Improvement



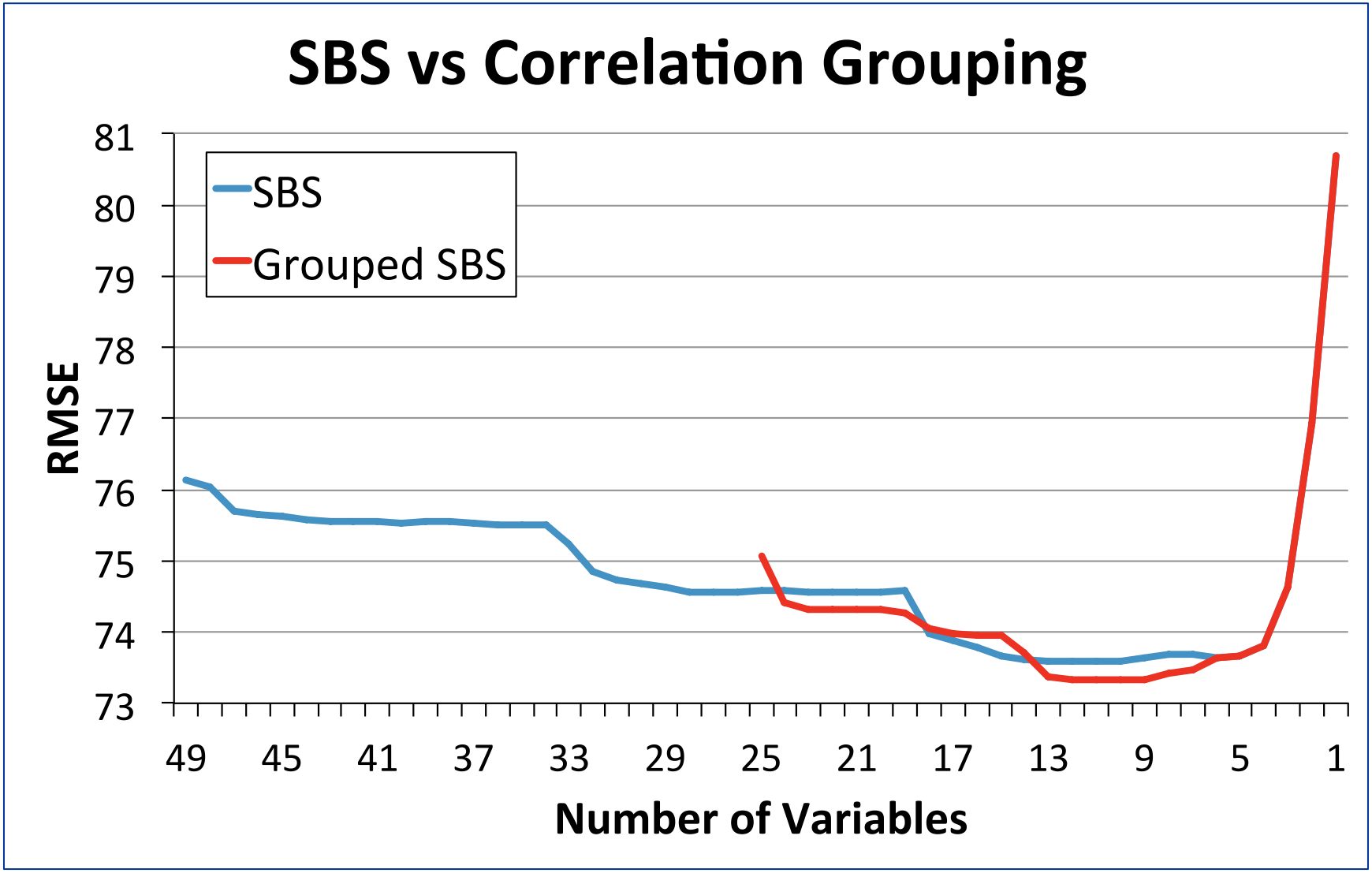
Correlation Grouping

- **Goal:**
  - Reduce iterations of SBS
  - Group highly correlated features
  - Select single variable from each group
  - Perform SBS on remaining subset
- **Implementation**
  - Use Breadth-First Search to search through correlation matrix
  - Group variables with correlation above correlation threshold parameter
  - Build prediction model for each group to select most critical variable

Correlation Grouping



Correlation Grouping Performance



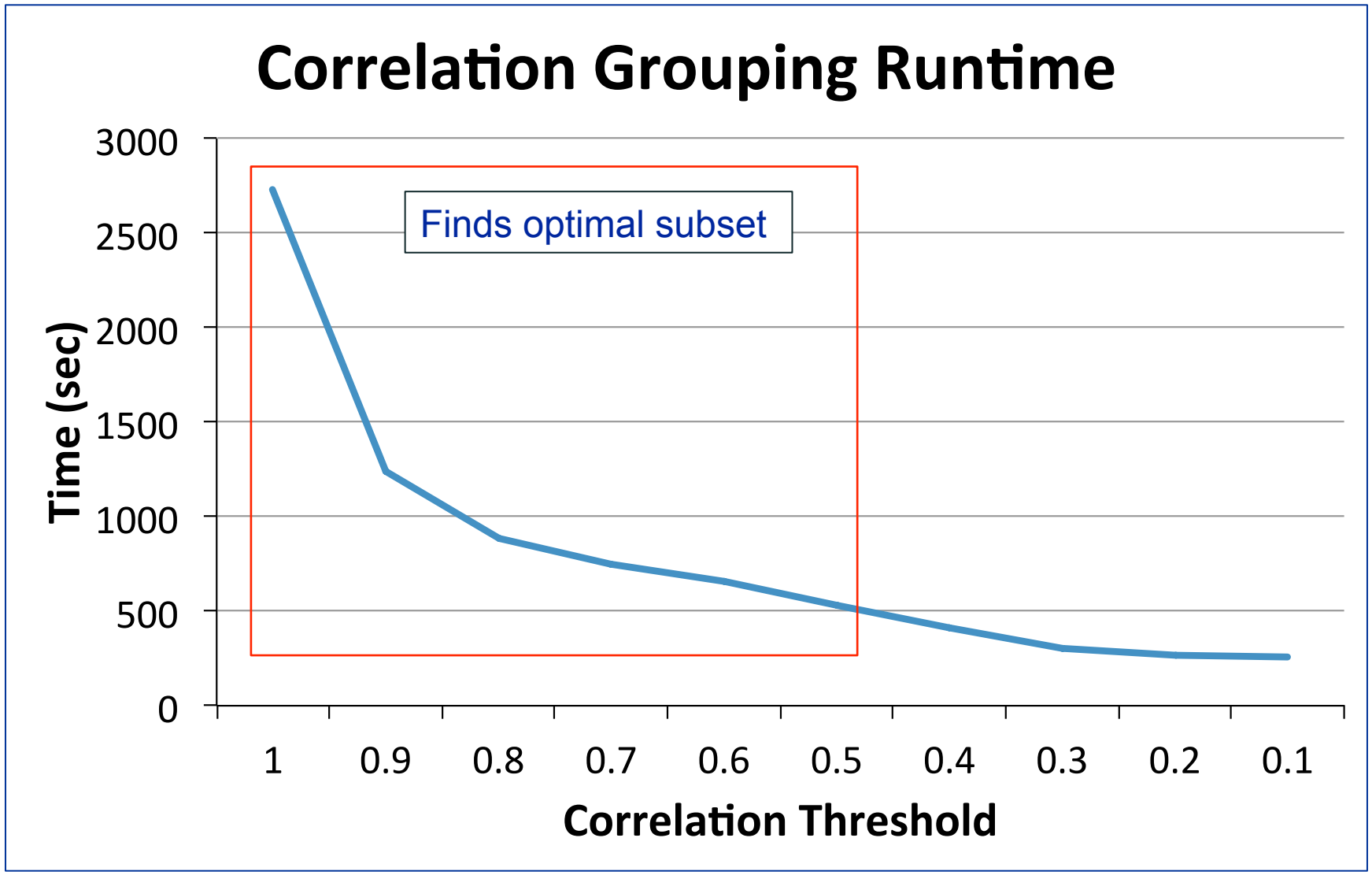
Correlation Grouping Performance

- **Improvement**
  - Improves serial runtime of SBS
  - Varies based on correlation of dataset
  - Eliminates most expensive models
  - Reduces runtime by 1/3rd on PTF dataset
- **Parallelization**
  - Test correlation groups in parallel
  - One core for each correlation group
  - Utilize 3 nodes as with SBS parallelization
  - Improvement depends on number of correlation groups

Correlation Threshold

- **Correlation Threshold Parameter**
  - Controls size of correlation groups
  - Balance accuracy with runtime improvement
  - Large groups eliminate too many features
  - Many small groups reduces runtime improvement
- **Selected Threshold**
  - Experimentally selected correlation = 0.8
  - Most improvement in runtime
  - Reduces runtime from 2727 sec to 888 sec

Correlation Grouping Threshold



Results

- SBS approximates exhaustive search
  - Sequential selection - does not add variables back
    - Not a significant drawback on this data
  - Identifies same optimal subset
  - Parallelized for large runtime improvement
- Correlation Grouping
  - Eliminates redundant variables quickly in parallel
  - Same results as SBS
  - Further runtime improvement