Let's Make Block Coordinate Descent Converge Faster: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence

Julie Nutini JULIE.NUTINI@GMAIL.COM

University of British Columbia

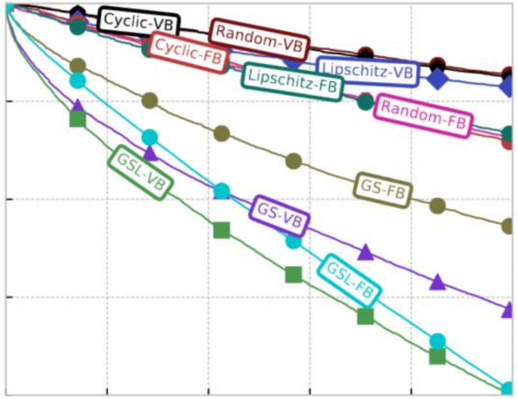
Issam Laradji ISSAMOU@CS.UBC.CA

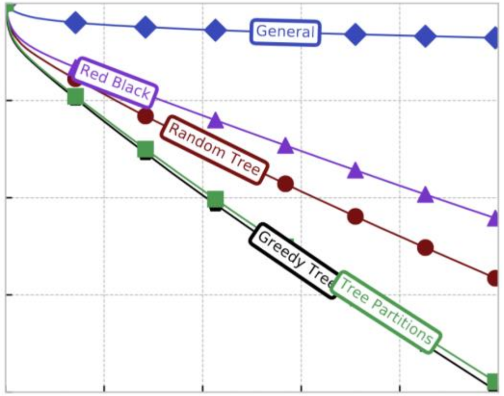
University of British Columbia, ServiceNow Research

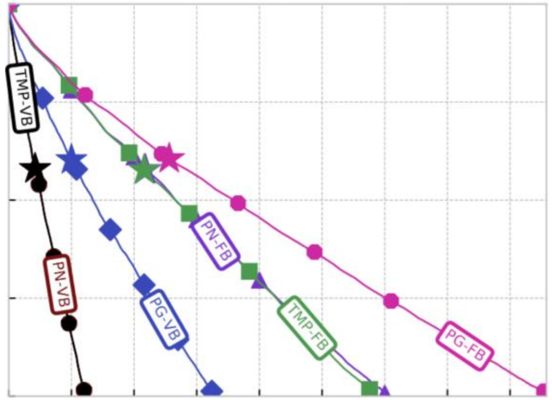
Mark Schmidt SCHMIDTM@CS.UBC.CA

University of British Columbia, Canada CIFAR AI Chair (Amii)



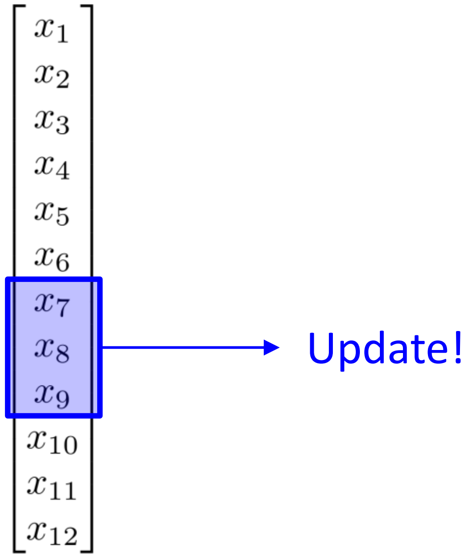







Block Coordinate Descent (BCD)

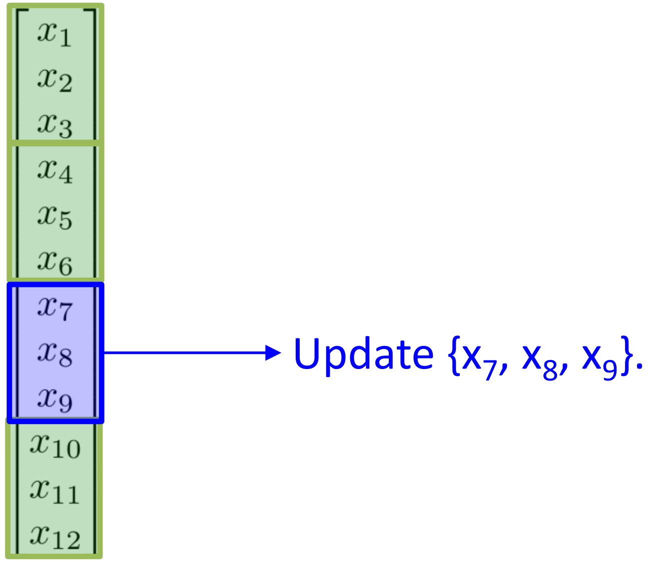
- Block coordinate descent (BCD) is a popular optimization algorithm in ML:
  - Each iterations selects and updates a block of variables to decrease objective.



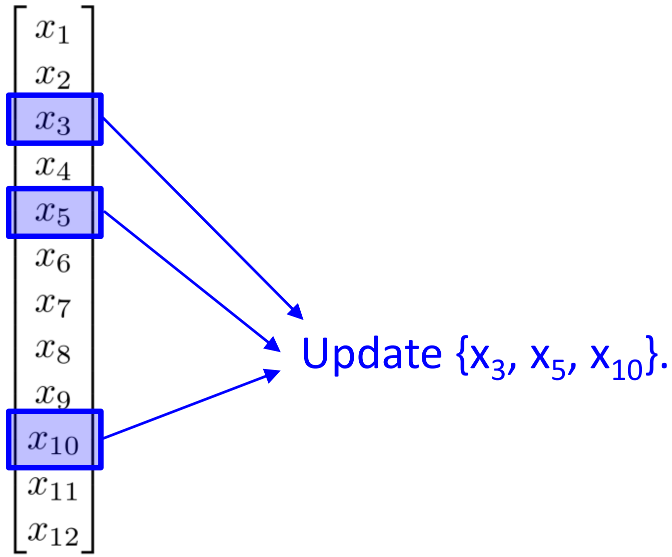
  - Can make similar progress to updating all variables.
    - But for some problems, updating block is much faster than updating all variables.
- There are many choices related to choosing the block and update.
- This work: ways to make BCD converge faster.
  - Gives faster algorithm when these not significantly increase iteration cost.

Fixed Blocks vs. Variable Blocks

- Fixed blocks:
  - Choose among a fixed partition.

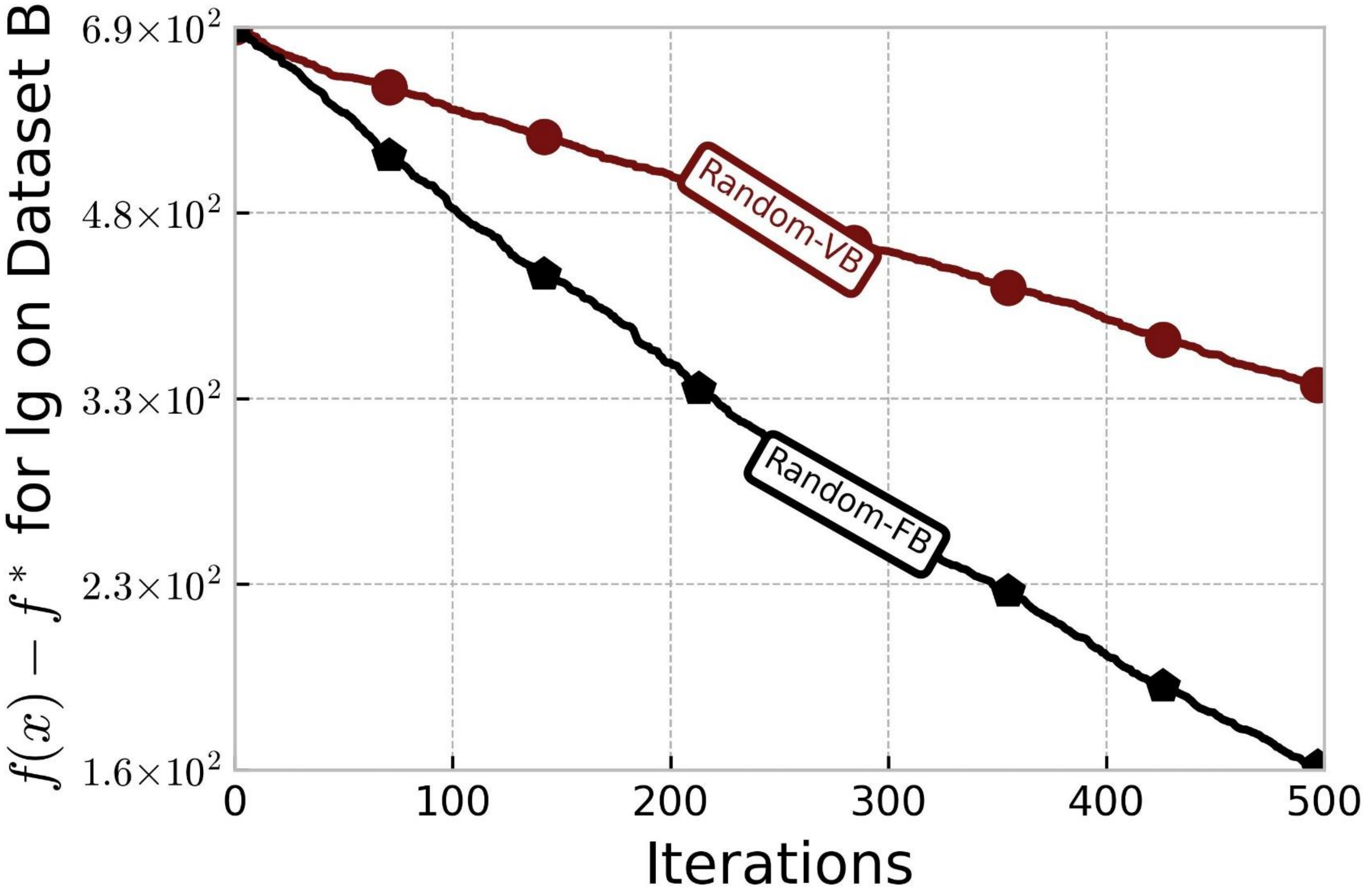


- Variable blocks:
  - Choose any subset of fixed size.



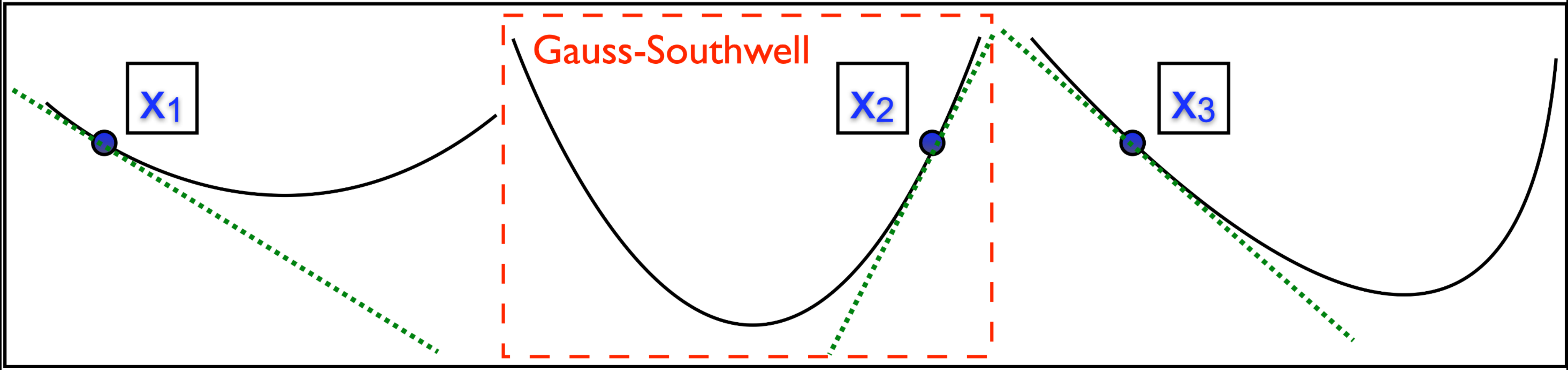
Fixed Blocks vs. Variable Blocks

- With random selection, **fixed blocks** seem to converge faster.



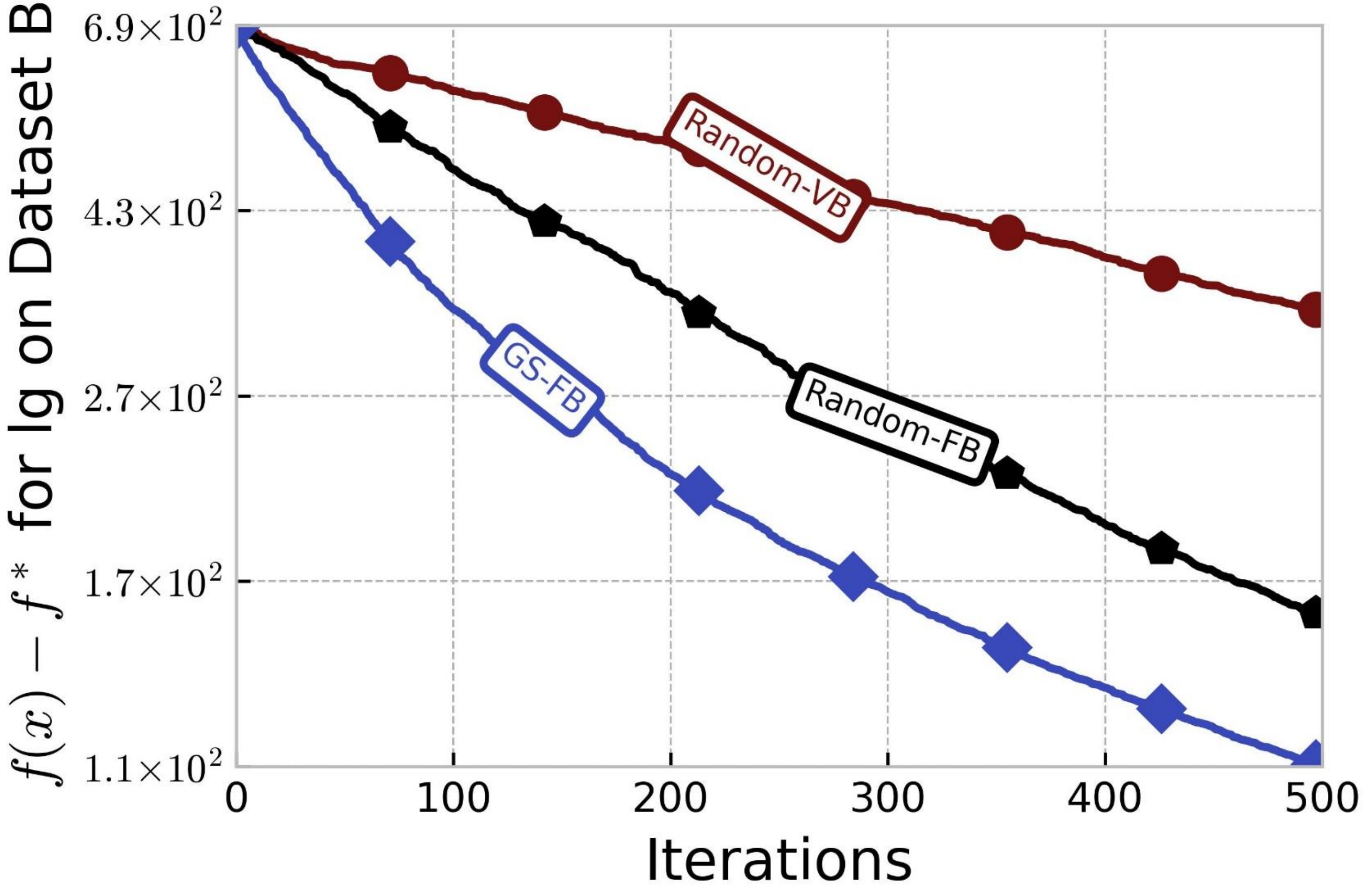
Random Rules vs. Greedy Gauss-Southwell Rule

- Random rule:
  - Sample among all possible blocks, each with equal probability.
- Greedy Gauss-Southwell (GS) rule:
  - Choose the block that has the largest gradient norm.



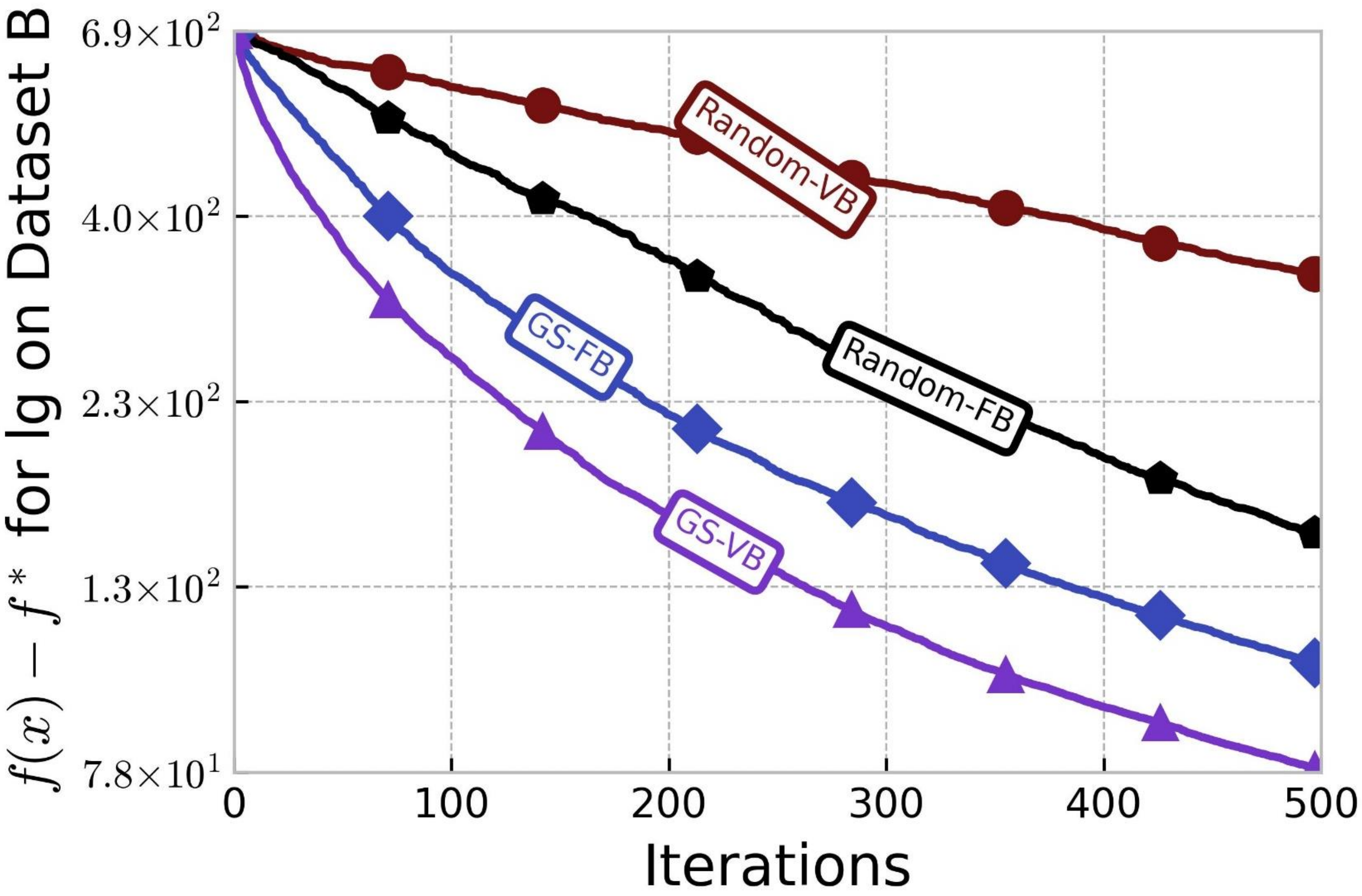
Random Rules vs. Greedy Gauss-Southwell Rule

- Greedy GS rule converges faster than random rule.



Random Rules vs. Greedy Gauss-Southwell Rule

- With greedy rule, variable blocks converge faster.



Direction of Update and Step Size.

- Gradient update:
  - Multiply gradient by a step size of 1/L (Lipshitz smoothness of block).

$$d^k = -\frac{1}{L_{b_k}} \nabla_{b_k} f(x^k)$$

- Matrix update:
  - Multiply gradient by upper bound on Hessian block with step size of 1.

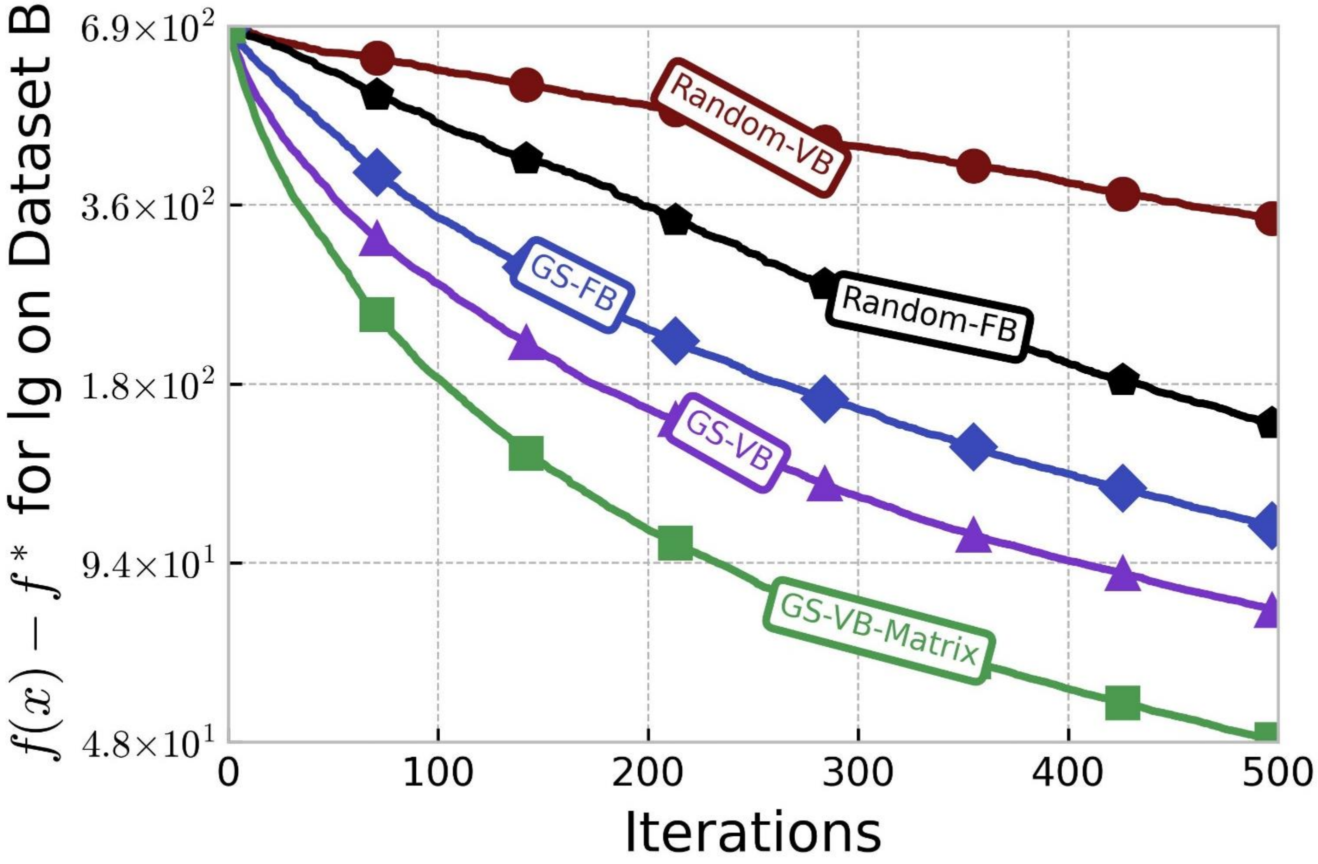
$$d^k = -(H_{b_k})^{-1} \nabla_{b_k} f(x^k)$$

- Newton update:
  - Multiply gradient by Hessian and a step size set by backtracking.

$$d^k = -\alpha_k \left(\nabla^2_{b_k b_k} f(x^k)\right)^{-1} \nabla_{b_k} f(x^k)$$

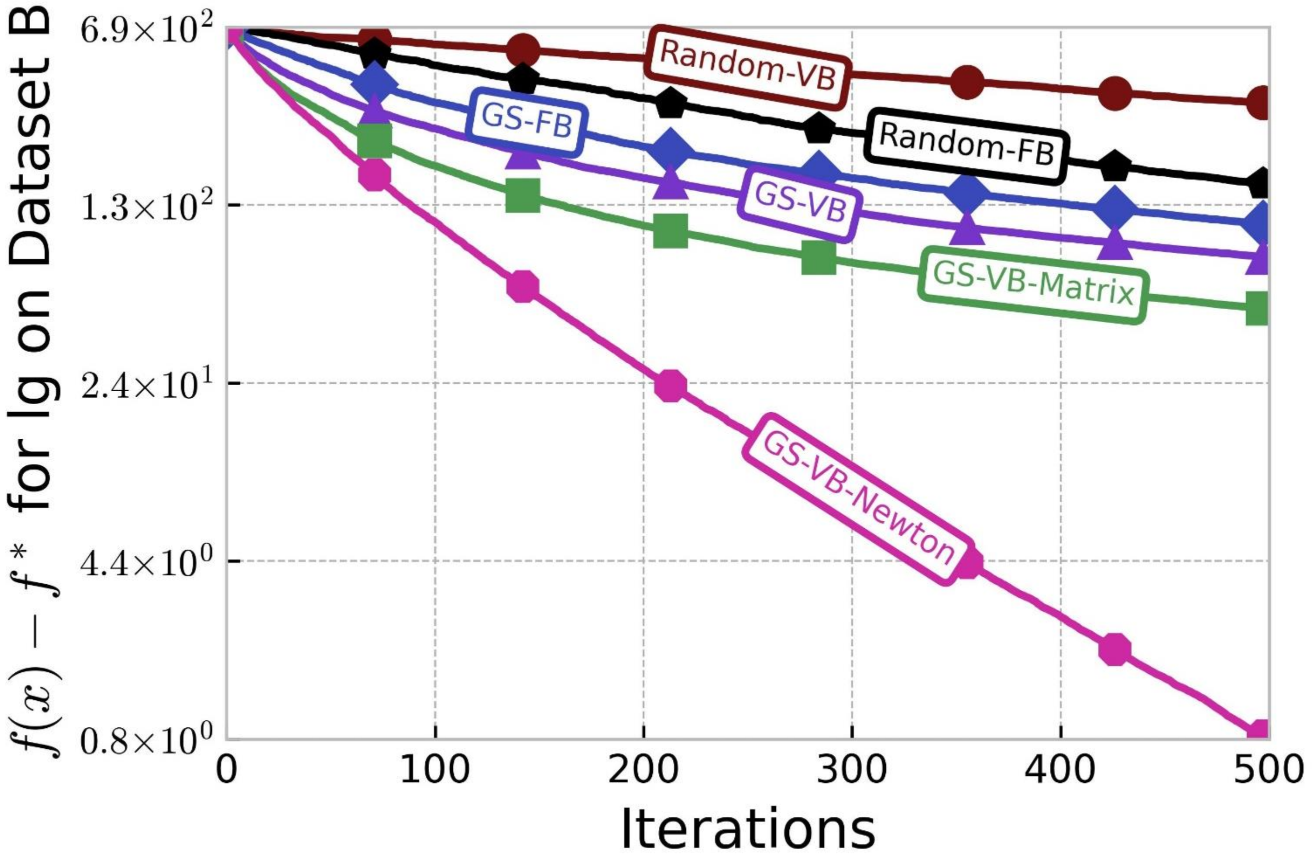
Direction of Update and Step Size

- Matrix update outperforms gradient update.



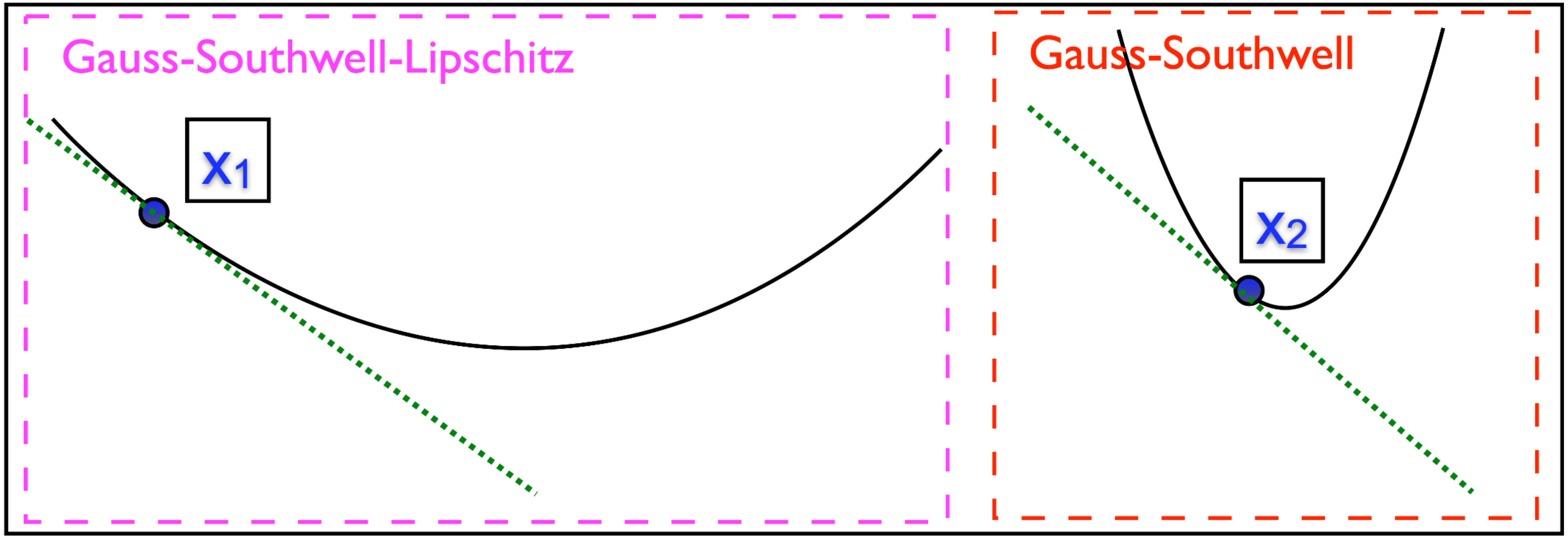
Direction of Update and Step Size

- Matrix update outperforms gradient update.
- But Newton converges faster than both.



Greedy Rules Incorporating Lipschitz Constants

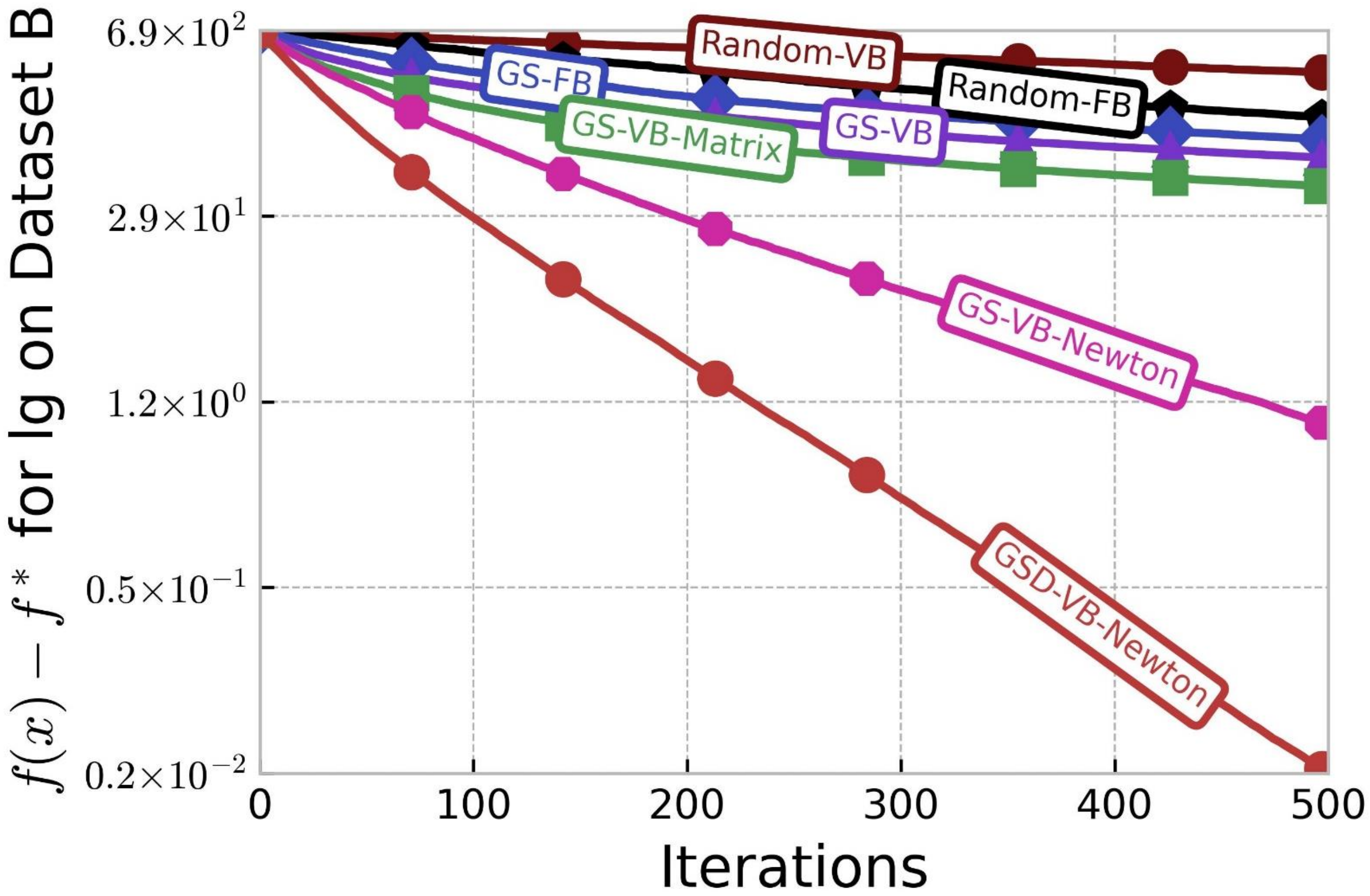
- Gauss-Southwell Lipschitz (GSL) rule:
  - Augment single-coordinate greedy with Lipschitz-continuity information.



  - We give several generalizations to block setting.

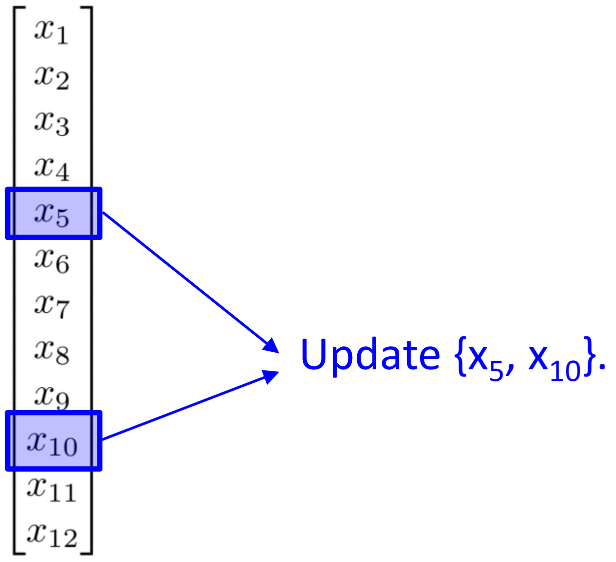
Greedy Rules Incorporating Lipschitz Constants

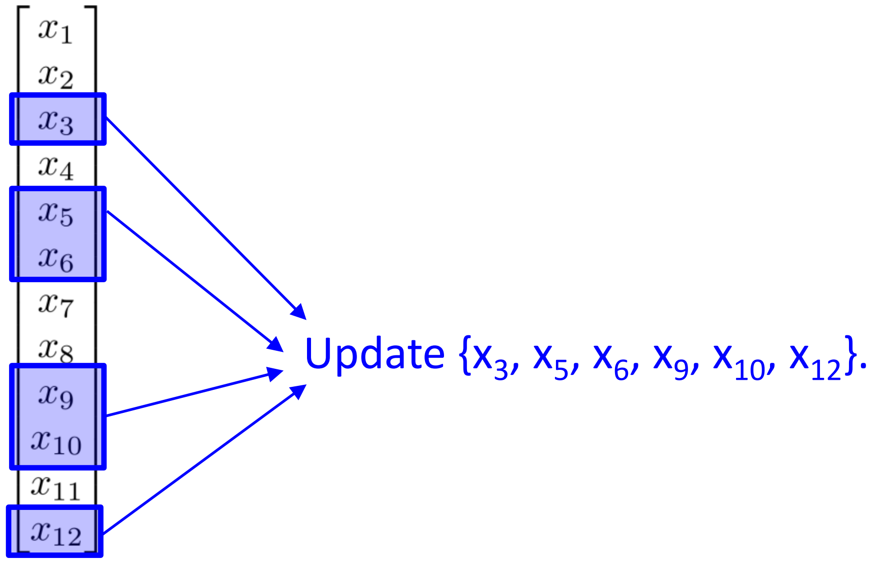
- All generalizations of GSL improved performance.



Block Size

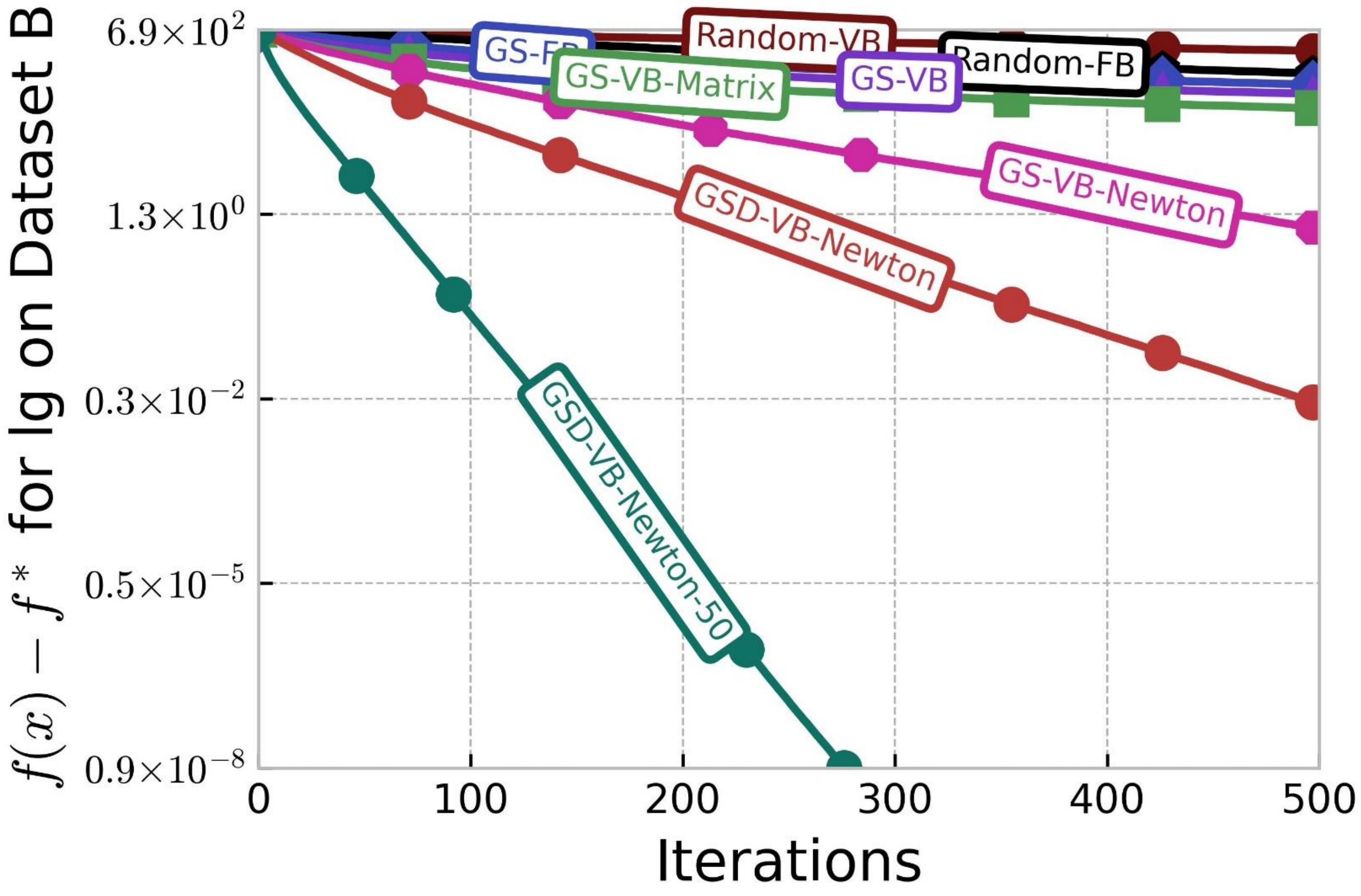
- Should we use small blocks or big blocks?





Block Size

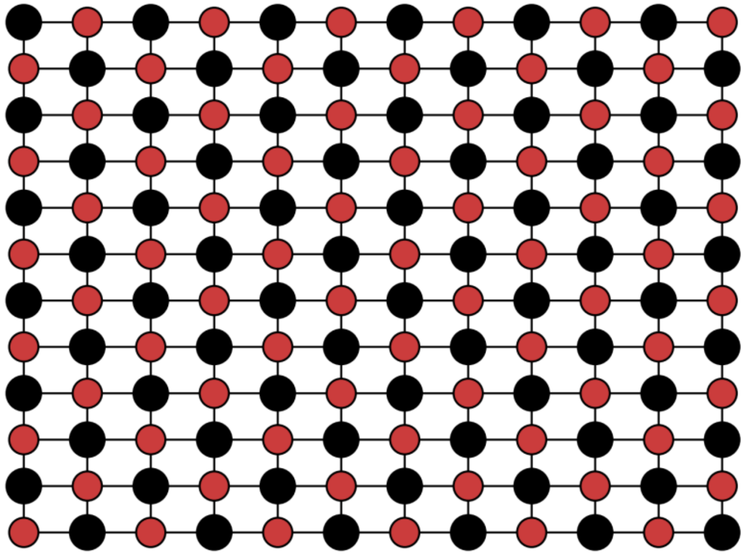
- Converges faster with bigger blocks.



Linear-Time Newton with Tree-Structured Blocks

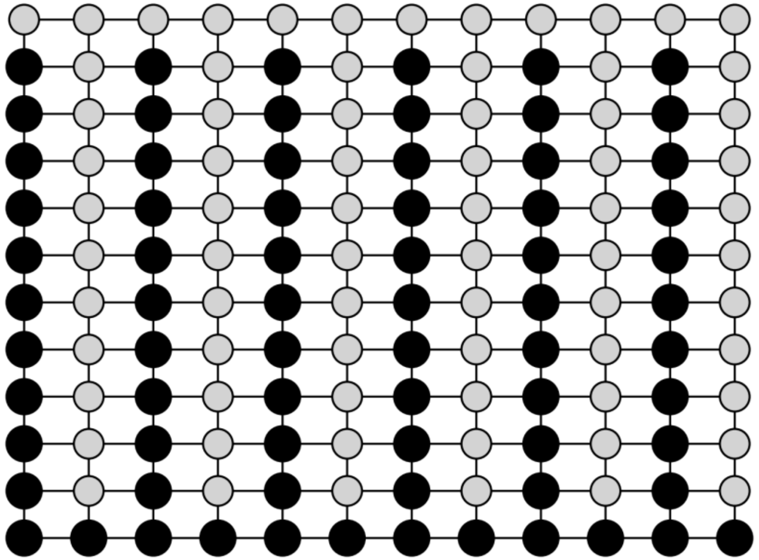
- Cost of Newton is cubic in block size.
  - But certain dependency structures allow linear-time updates.

Colouring Partition



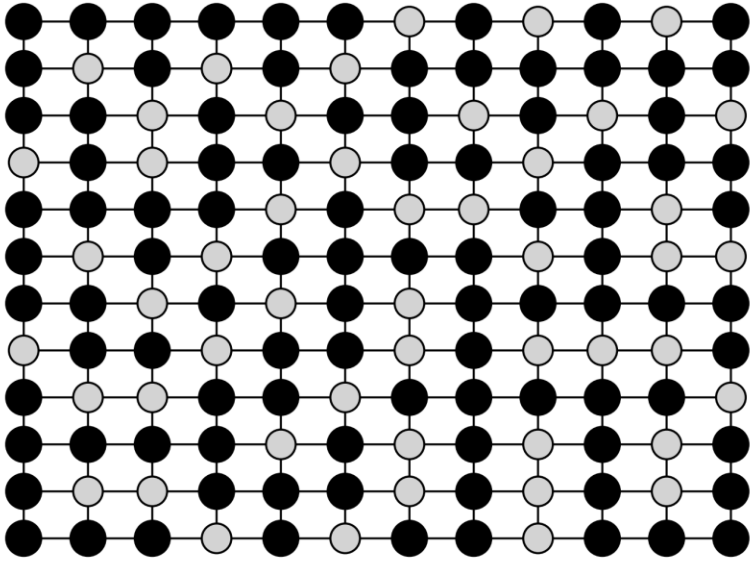
(block size of n/2 above)

Fixed Tree Partition



(block size of n/2 above)

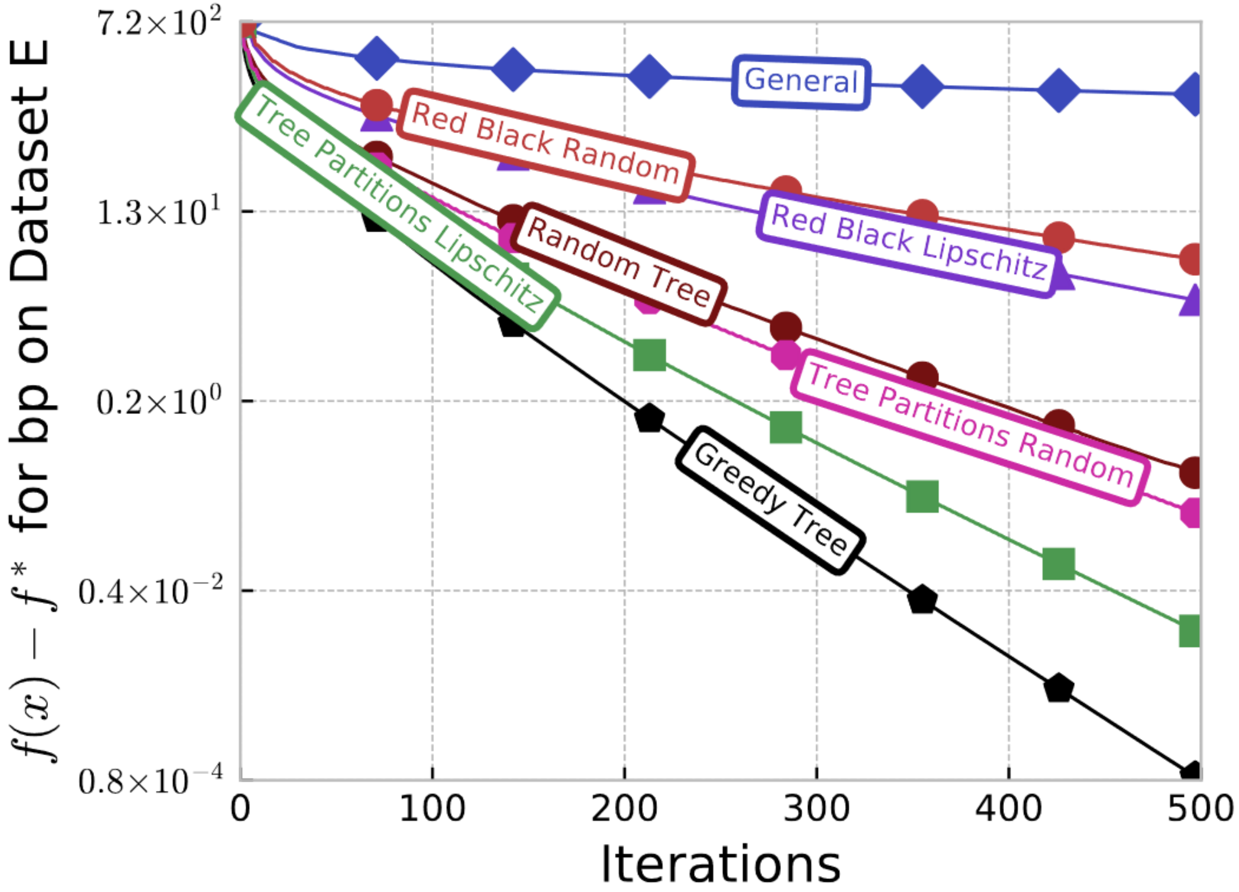
Variable Tree Partition



(block size of ≈2n/3 above)

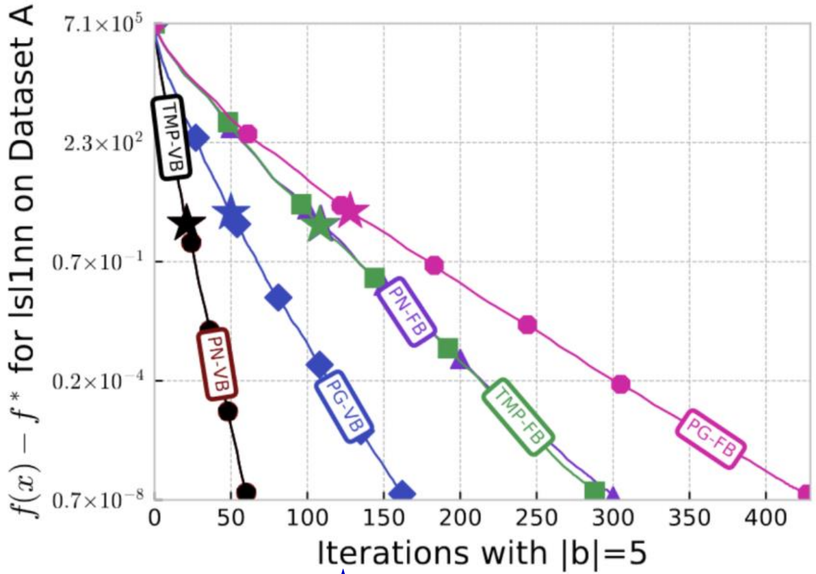
Linear-Time Newton with Tree-Structured Blocks

- Colouring can be faster than using small blocks.
  - But trees converge faster than colouring.



Bound Constraints and Non-Smooth Regularizers

- We often use BCD with bound constraints or L1-regularizers.
  - Gradient updates can be replaced with projected-gradient (cheap).
  - Newton updates can be replaced with projected-Newton (expensive).
    - **Two-metric projection** allows Newton-like updates without extra cost.



  - BCD identifies active variables (★) after a finite number of iterations.
    - Superlinear rate with greedy rules, large-enough variable blocks, PN/TMP updates.

Let's Make Block Coordinate Descent Converge Faster: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence

Come see us on Tuesday at Poster #109.