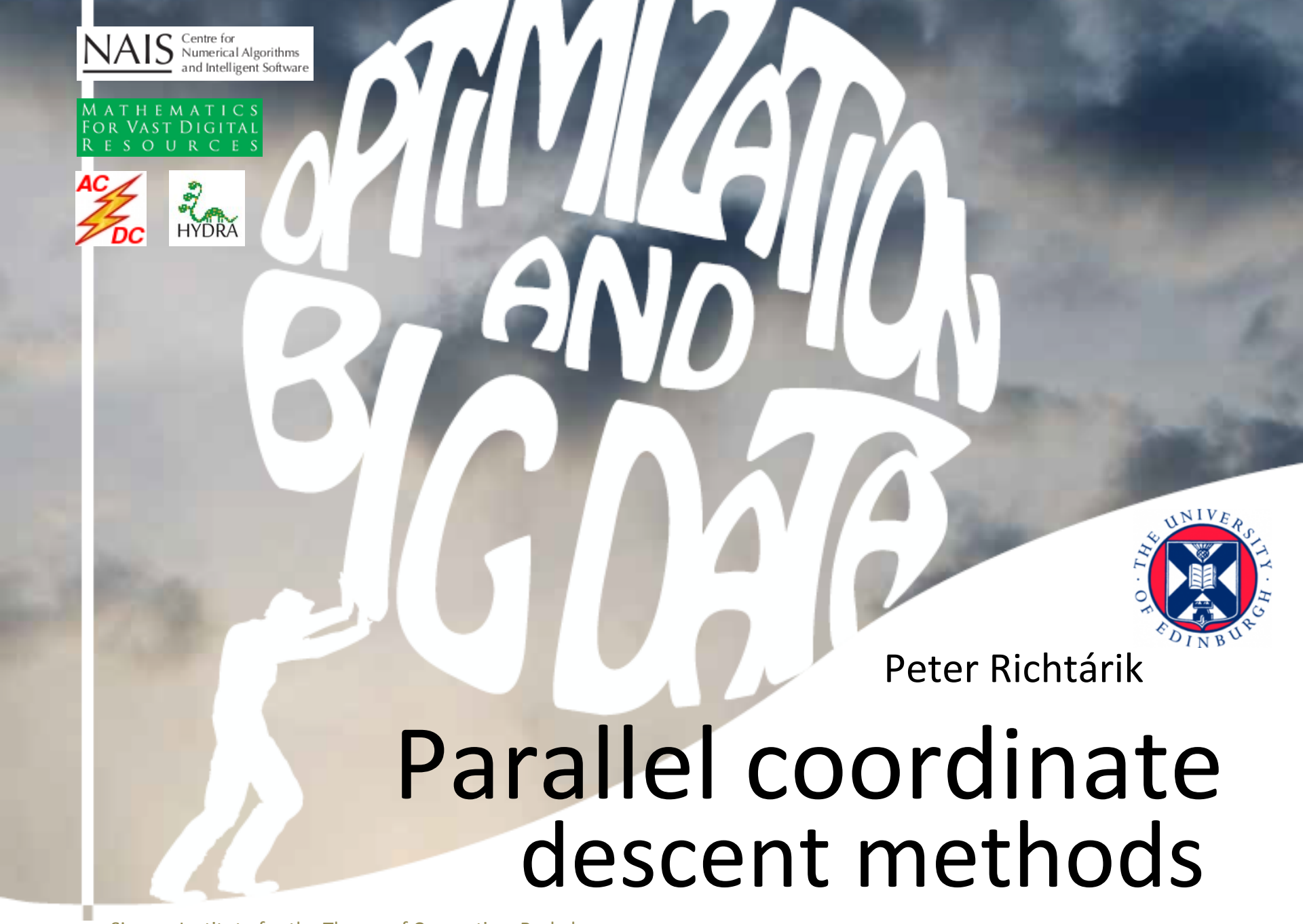


NAIS Centre for Numerical Algorithms and Intelligent Software

MATHEMATICS FOR VAST DIGITAL RESOURCES

AC DC

HYDRA

THE UNIVERSITY OF EDINBURGH

Peter Richtárik

Parallel coordinate descent methods

Simons Institute for the Theory of Computing, Berkeley
Parallel and Distributed Algorithms for Inference and Optimization, October 23, 2013

ALAN COCHRANE TAXIS
38 HARRYSMUIR GARDENS
THANK YOU

TERMINAL ID: ****3577
MERCHANT ID: **********21661

MASTERCARD
**********6875 ICC
PAN SEQ NO: 12
AID: A0000000041010

SALE
AMOUNT £100009.96

*** CUSTOMER COPY ***

TRANSACTION VOID

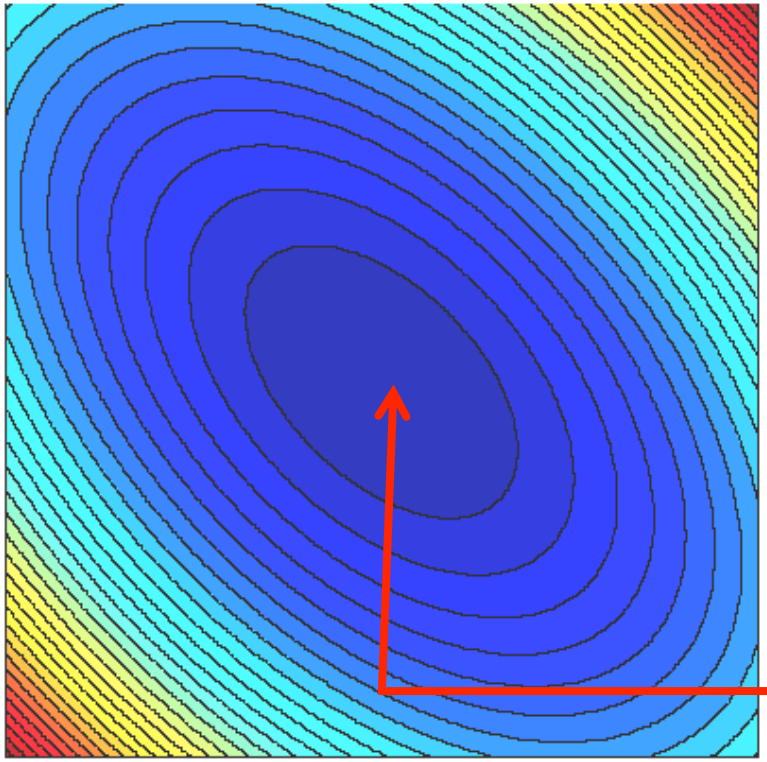
DATE: 05/03/13 TIME: 16:00

Randomized Coordinate Descent in 2D

2D Optimization

Contours of a function

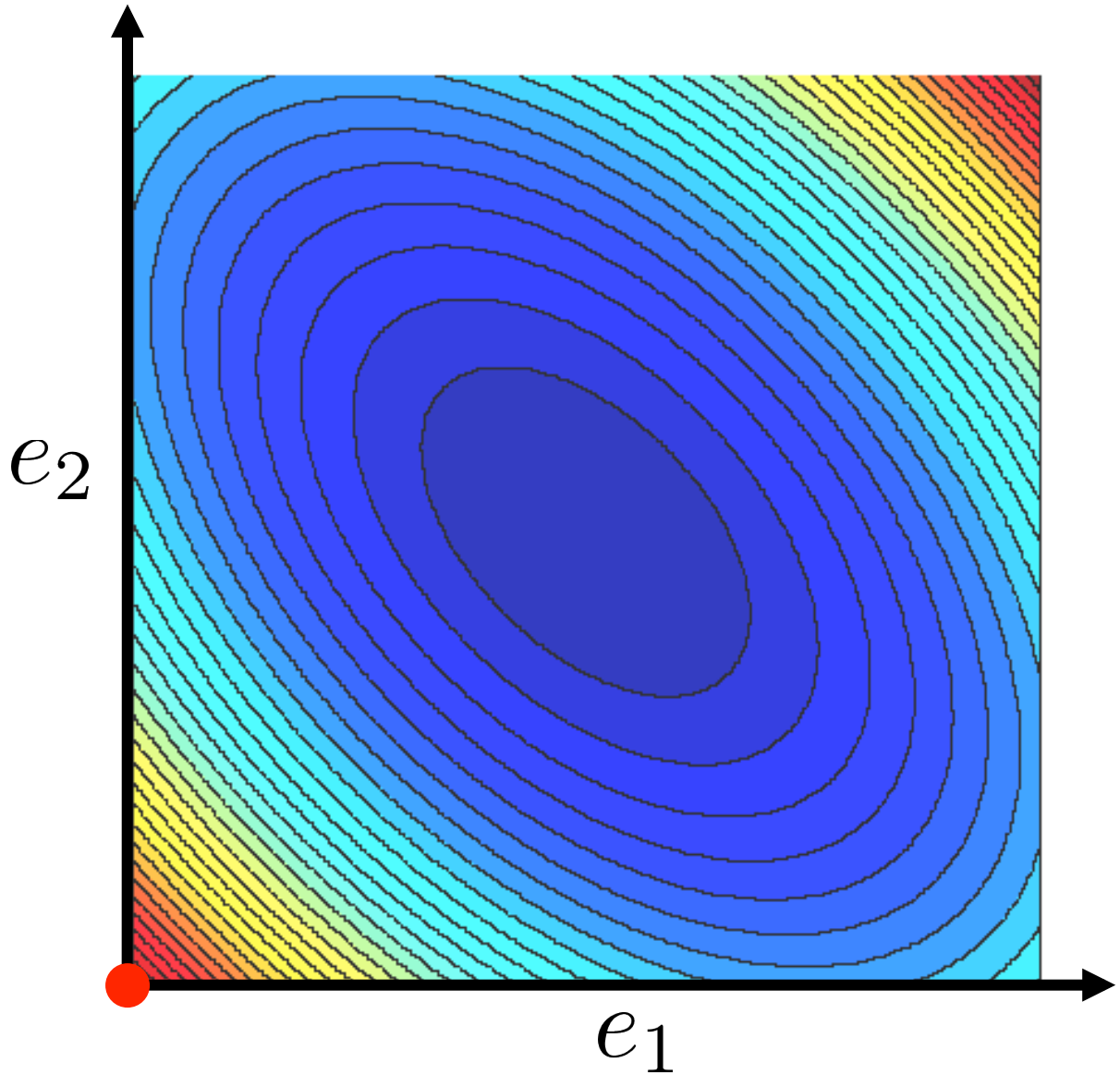
$$F : \mathbf{R}^2 \to \mathbf{R}$$

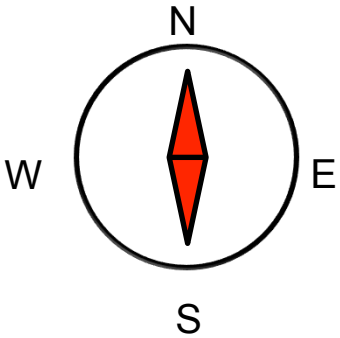


Goal:

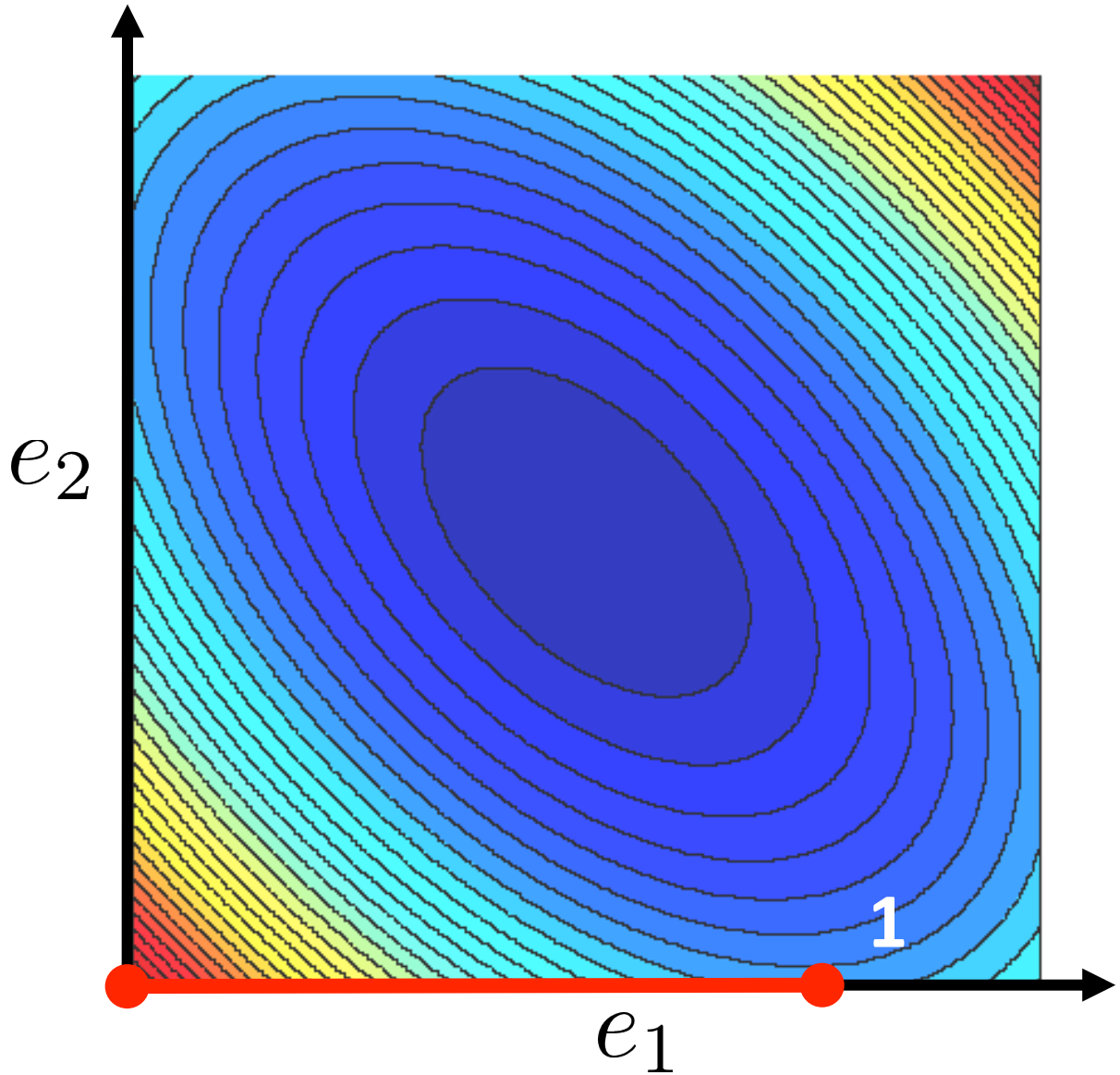
Find the minimizer of F

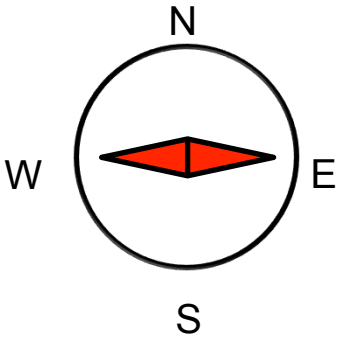
Randomized Coordinate Descent in 2D



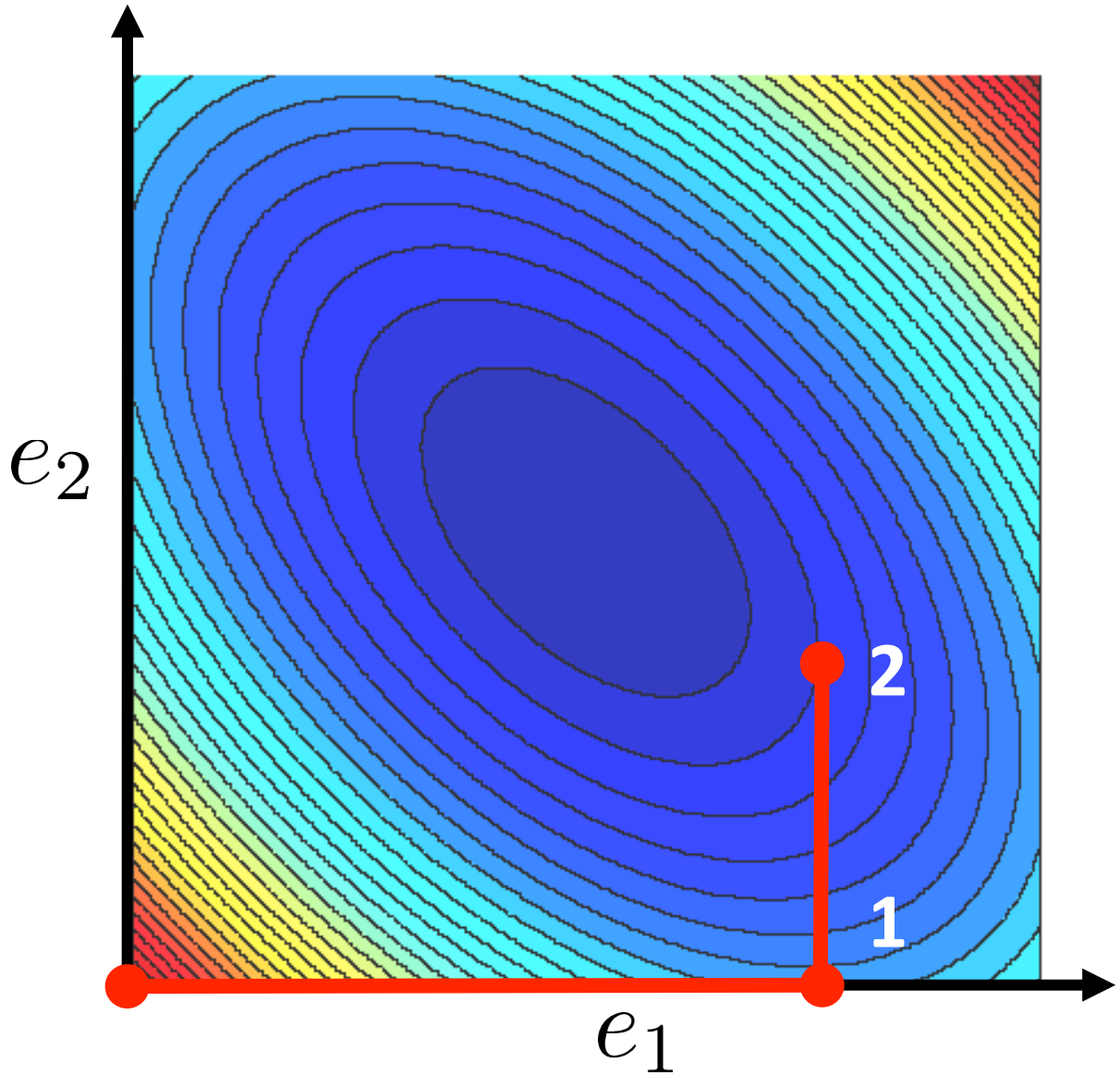


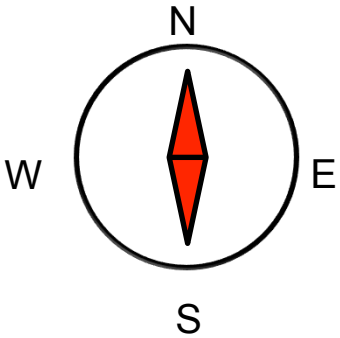
Randomized Coordinate Descent in 2D



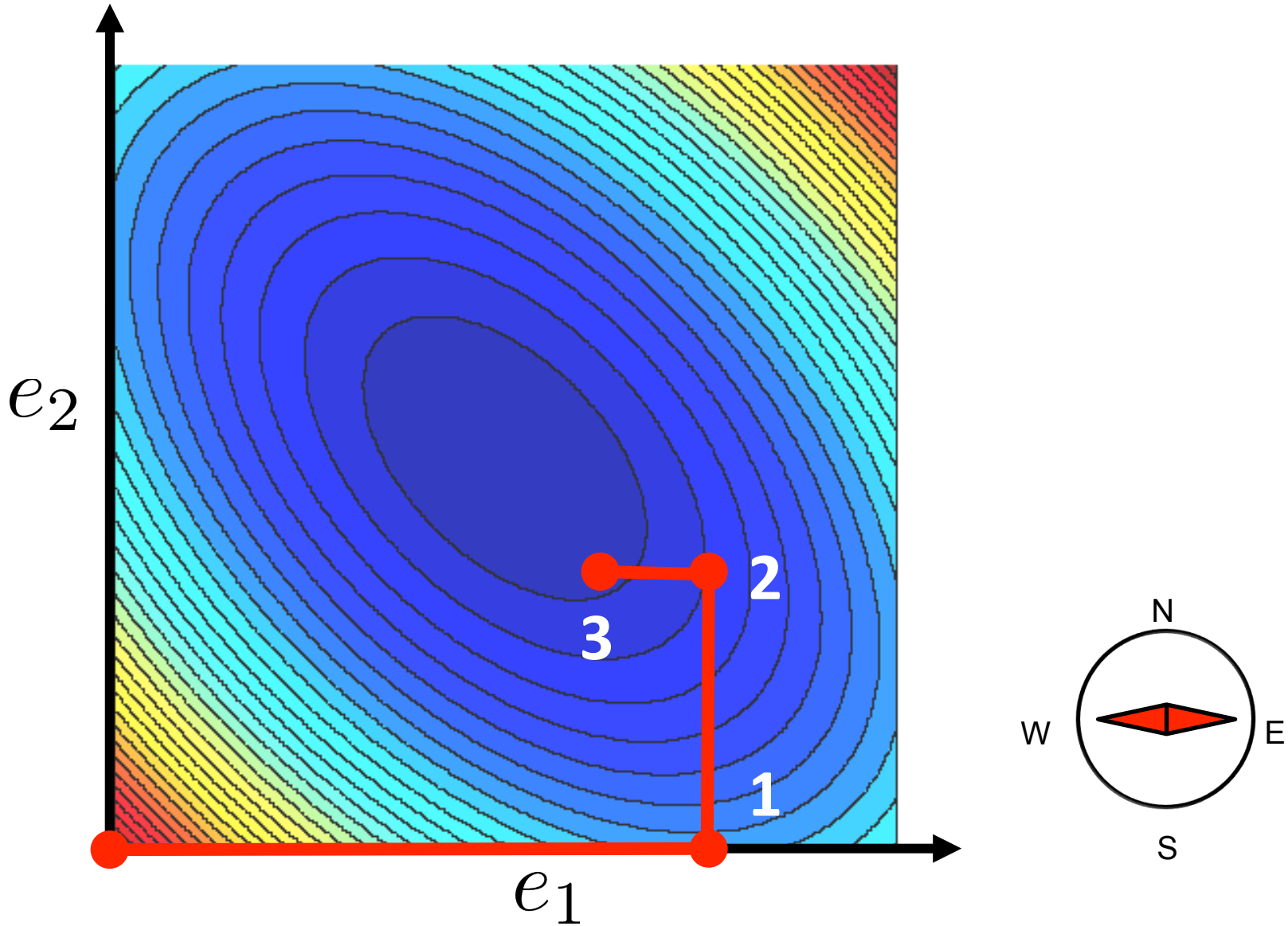


Randomized Coordinate Descent in 2D

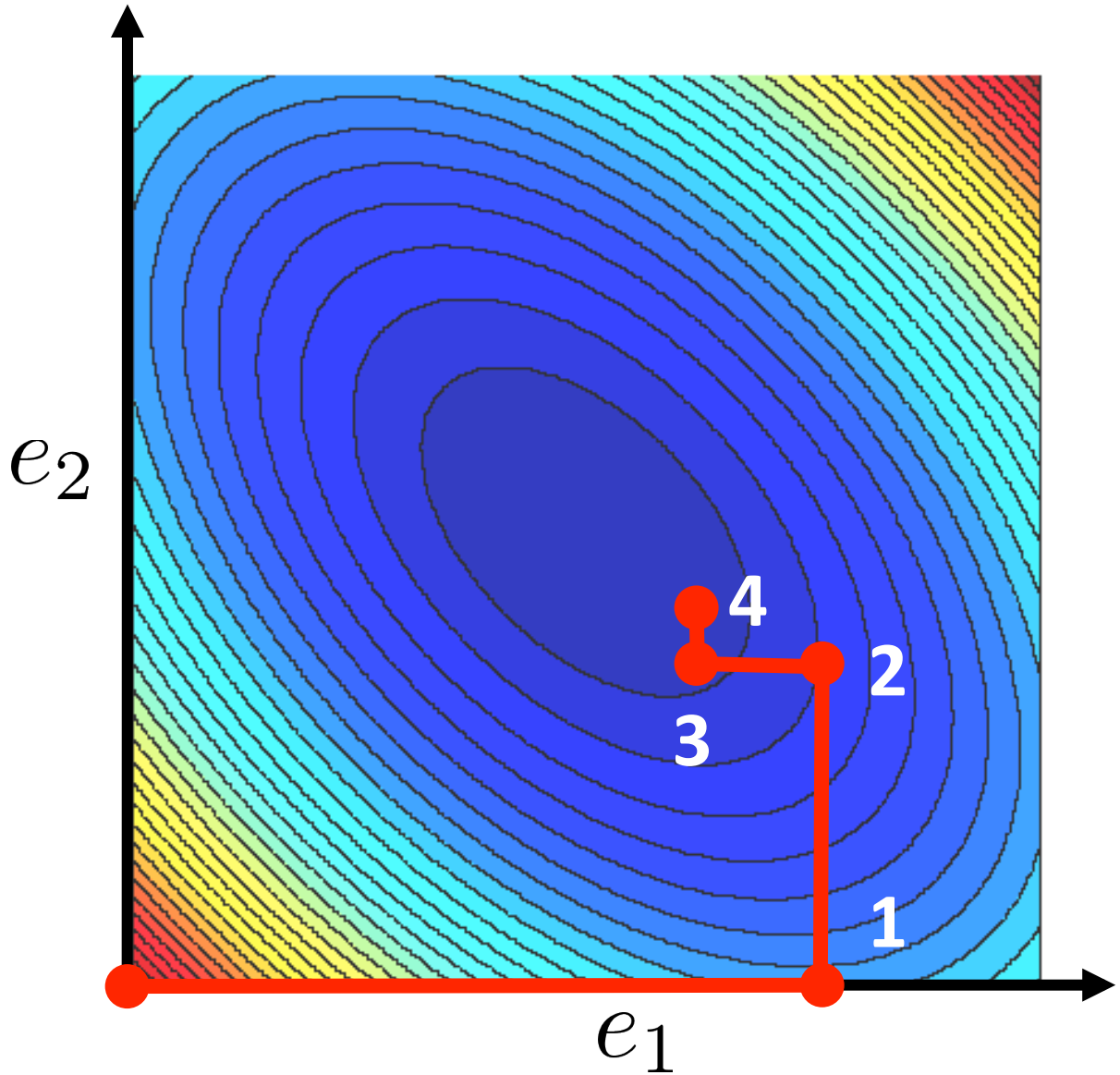


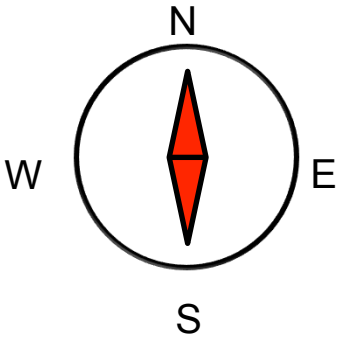


Randomized Coordinate Descent in 2D

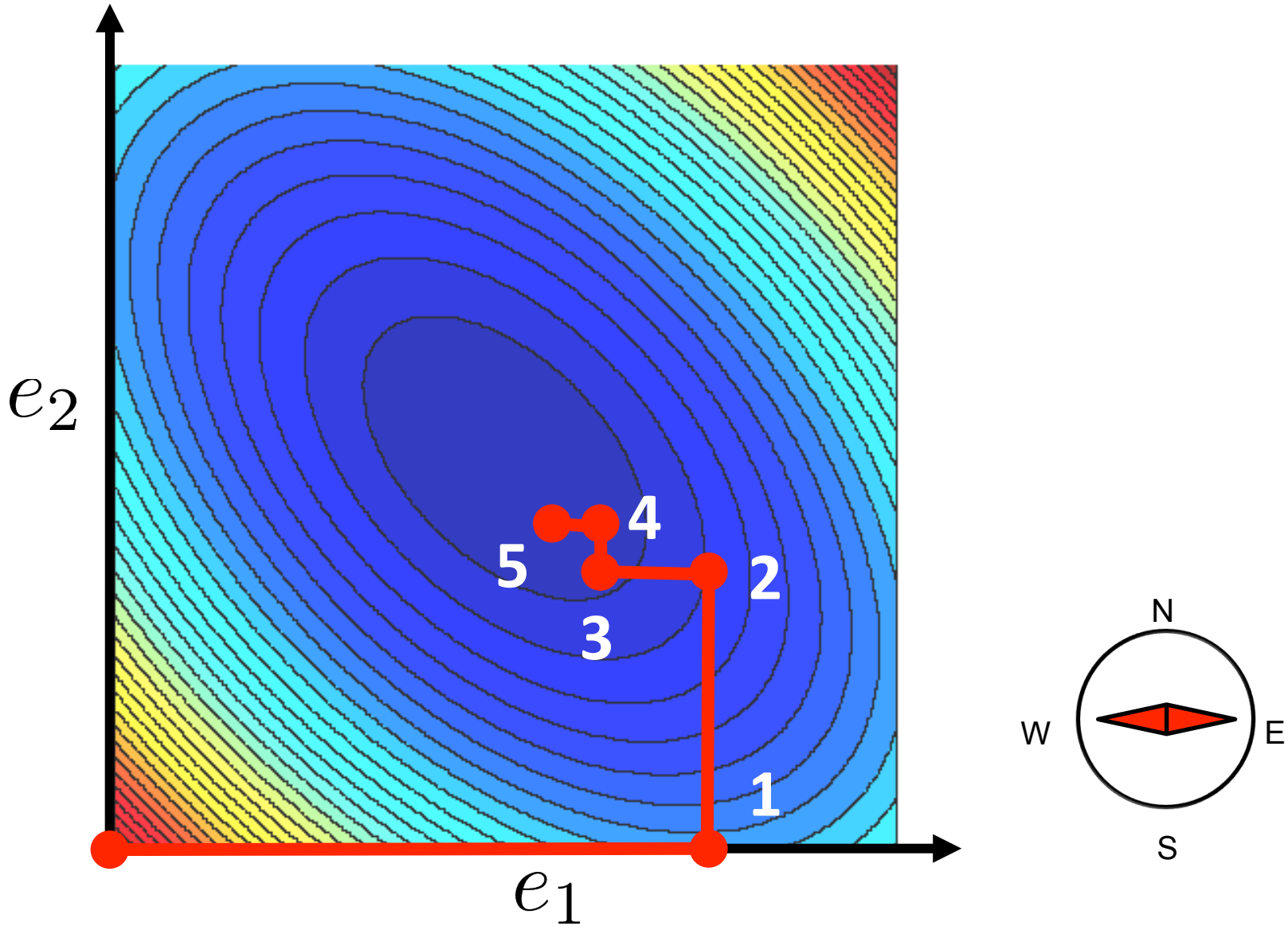


Randomized Coordinate Descent in 2D

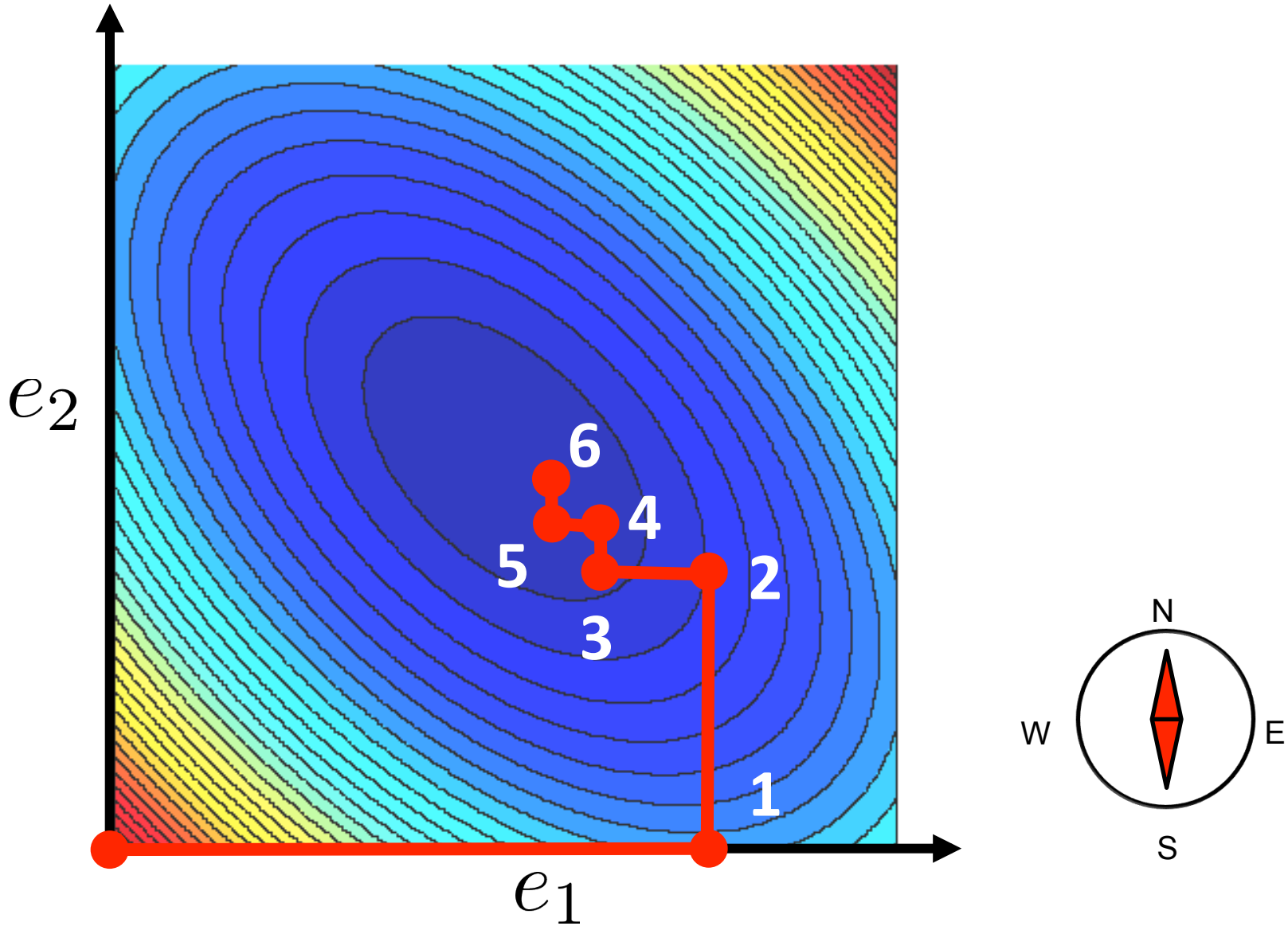




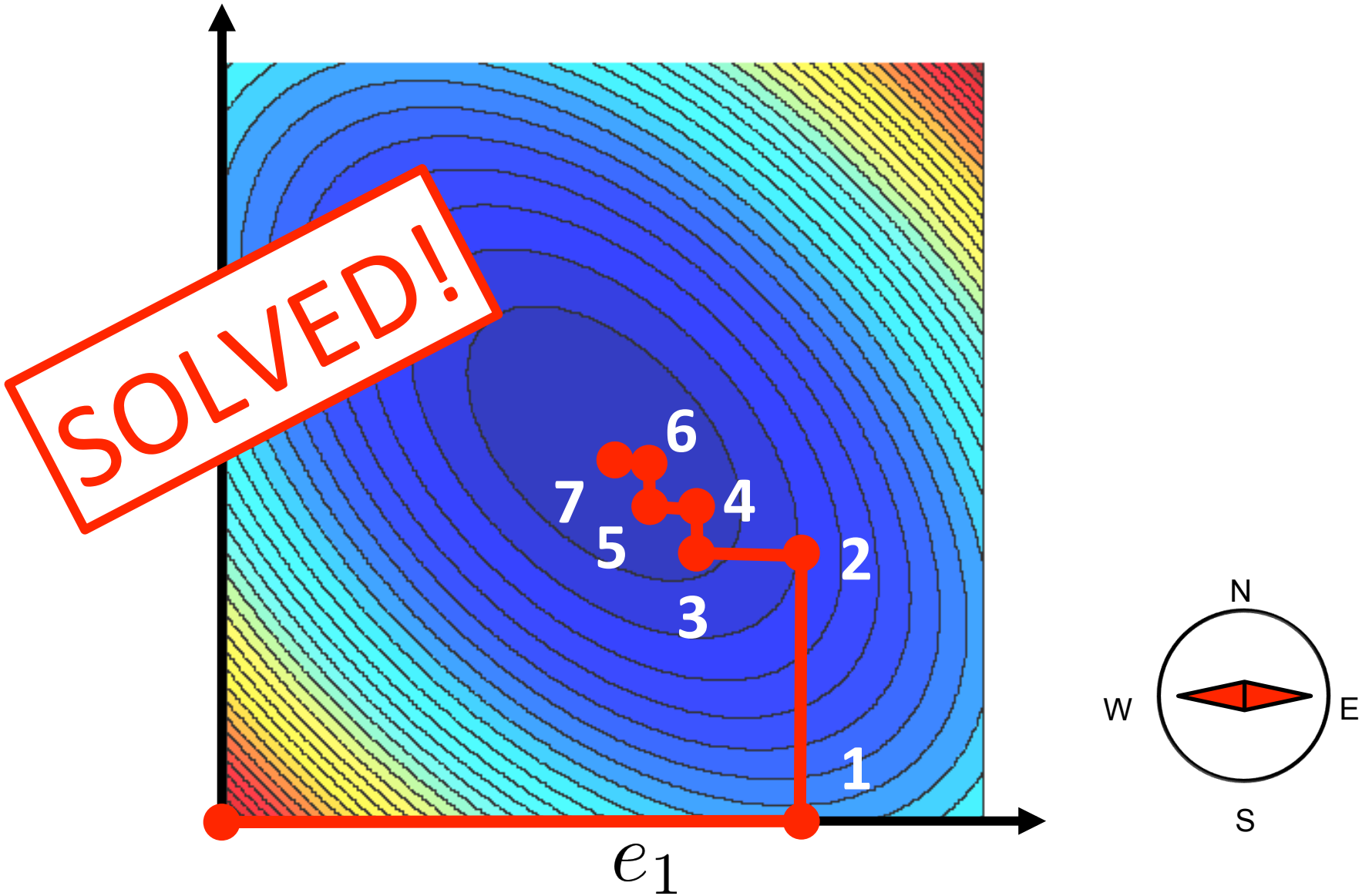
Randomized Coordinate Descent in 2D



Randomized Coordinate Descent in 2D



Randomized Coordinate Descent in 2D



Convergence of Randomized Coordinate Descent

In $\mathbb{R}^n$, randomized coordinate descent with uniform probabilities needs

$$O\left(n \times \cancel{\xi(\epsilon)}\right) \text{ iterations}$$

Strongly convex *F*

$$\xi(\epsilon) = \mathrm{lo}$$

Focus on *n*
(big data = big *n*)

Smooth or 'simp ... *F*

'difficult' nonsmooth *F*

Parallelization Dream

Serial

(1 coordinate per iteration)

$$O\left(n \times \xi(\epsilon)\right)$$

Parallel

(τ coordinates per iteration)

$$O\left(\frac{n}{\tau} \times \xi(\epsilon)\right)$$

What do we actually get?

$$O\left(\frac{n\beta}{\tau} \times \xi(\epsilon)\right)$$

WANT

$$\beta = O(1)$$

Depends on to what extent we can **add up** individual updates, which depends on the **properties of *F*** and **the way coordinates are chosen** at each iteration

How (not) to Parallelize Coordinate Descent

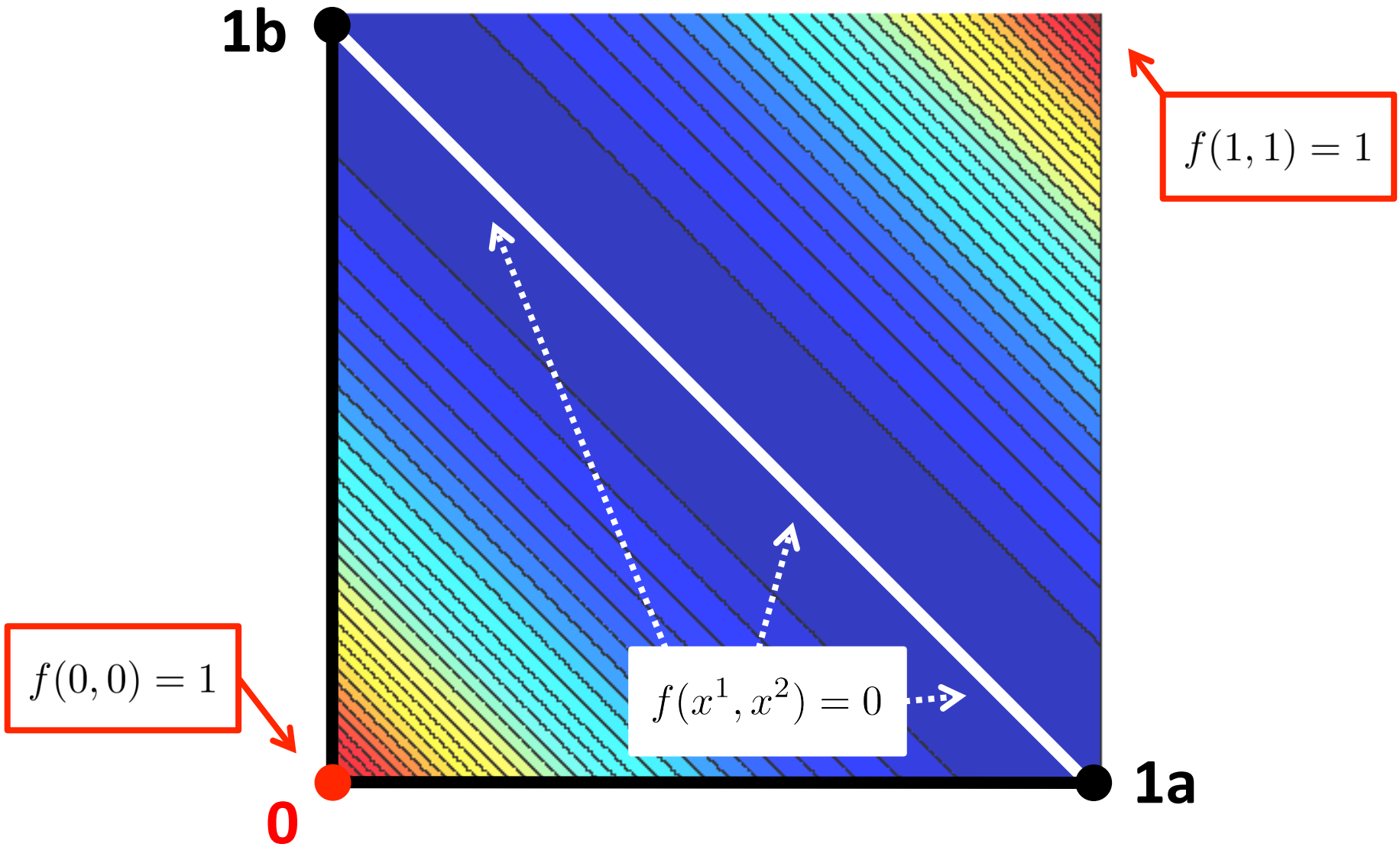
“Naive” parallelization

Do the same thing as before, but

for **MORE or ALL** coordinates
&
ADD UP the updates

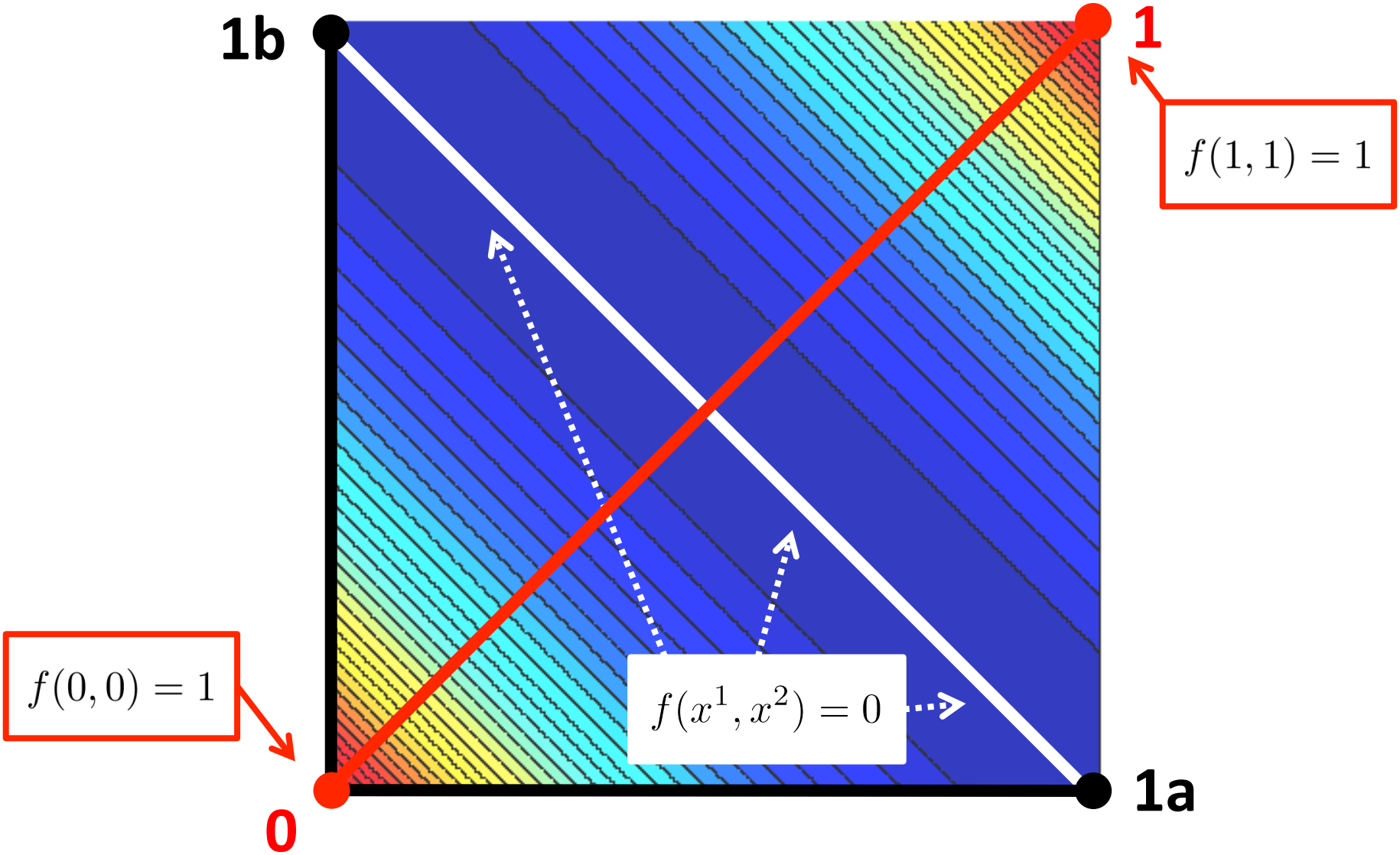
Failure of naive parallelization

$$f(x^1, x^2) = (x^1 + x^2 - 1)^2$$

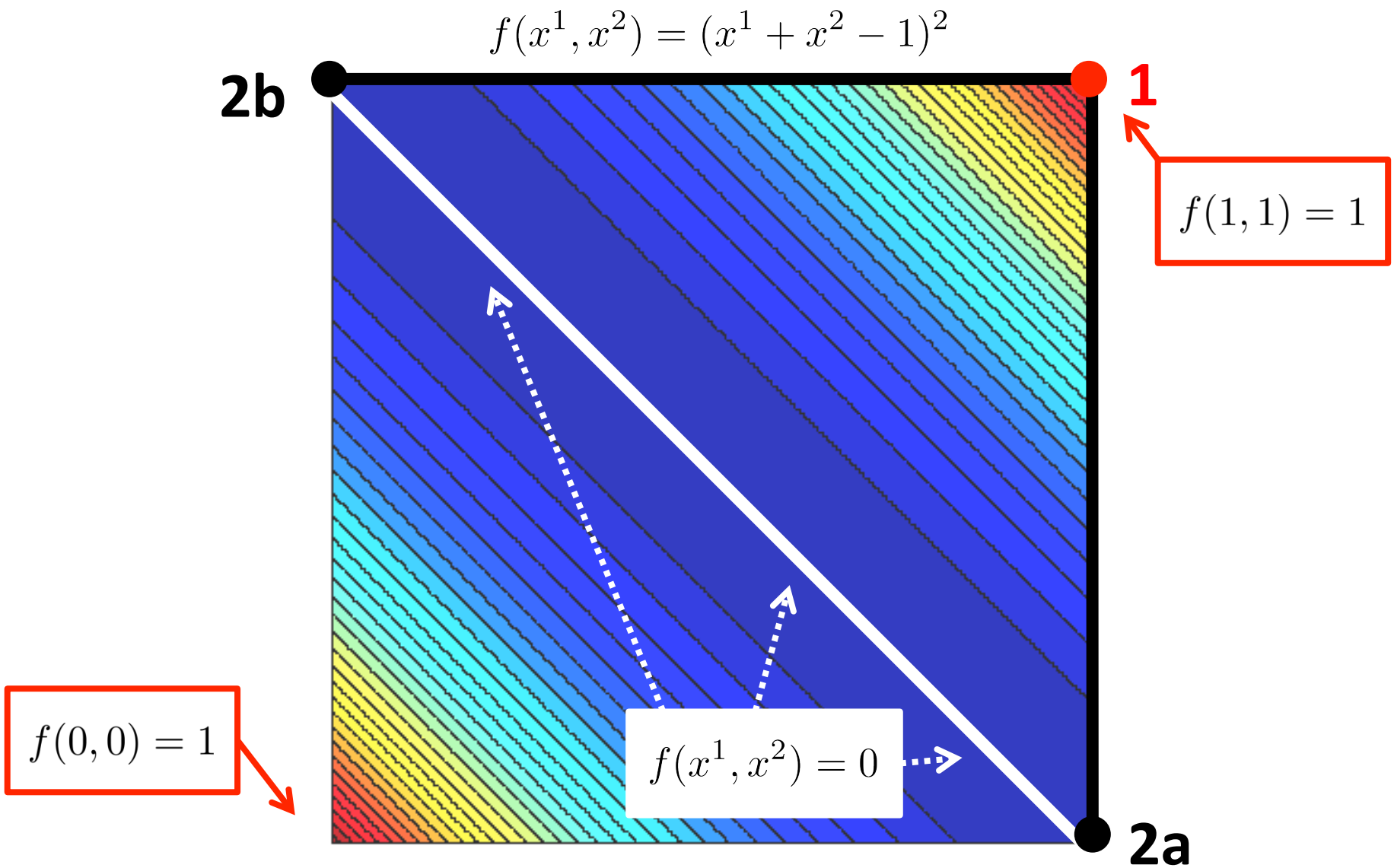


Failure of naive parallelization

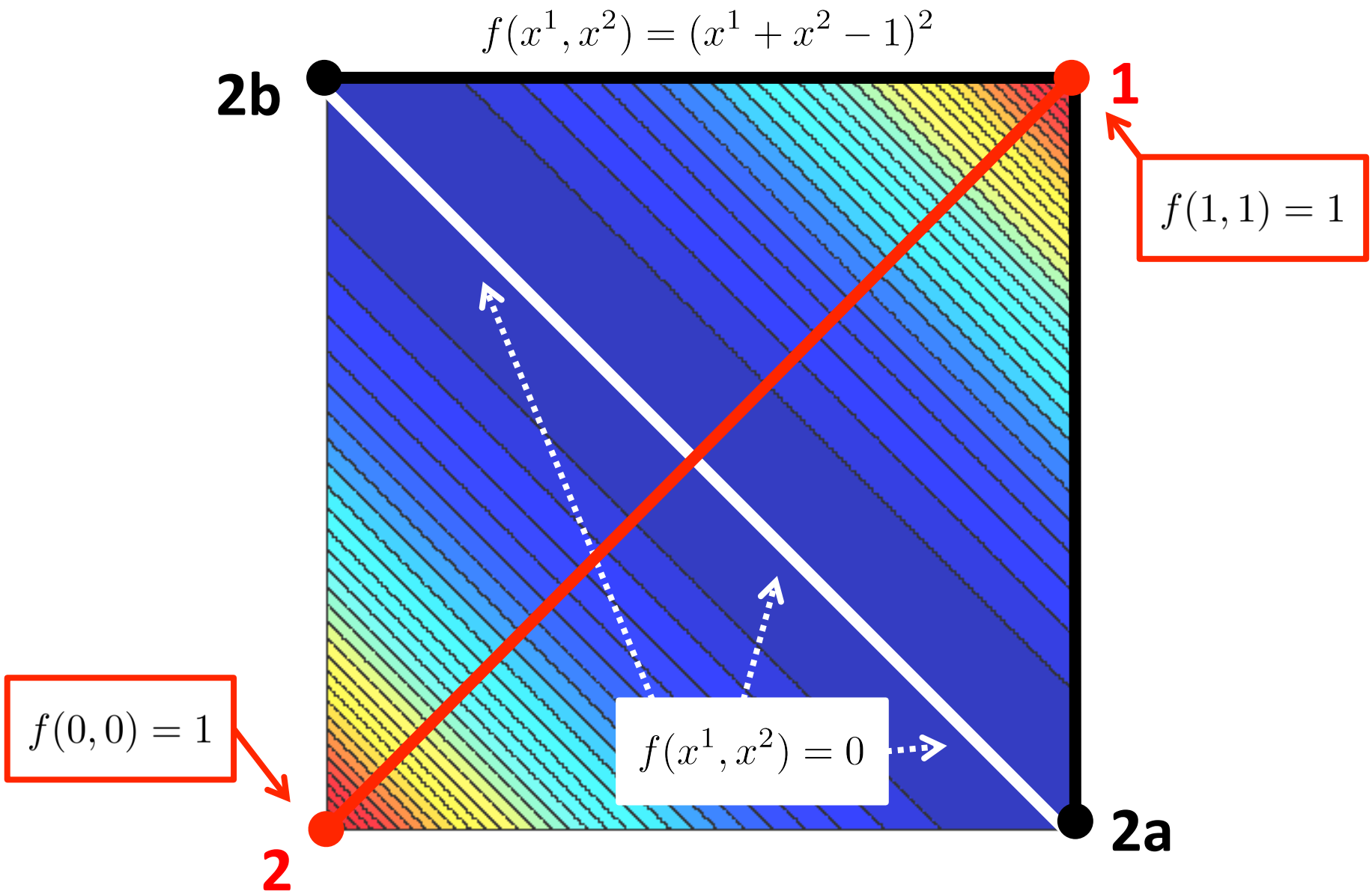
$$f(x^1, x^2) = (x^1 + x^2 - 1)^2$$



Failure of naive parallelization



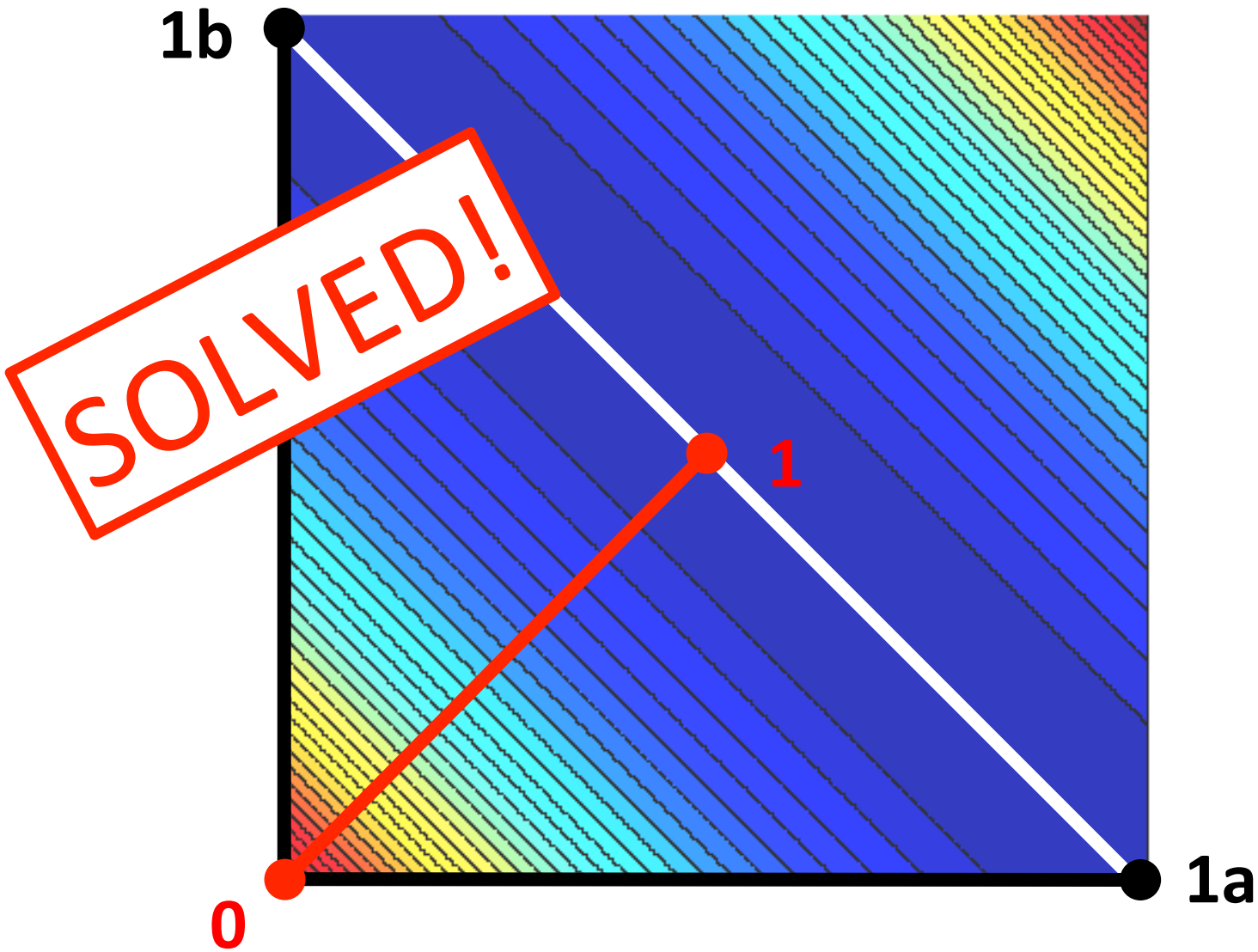
Failure of naive parallelization



Failure of naive parallelization

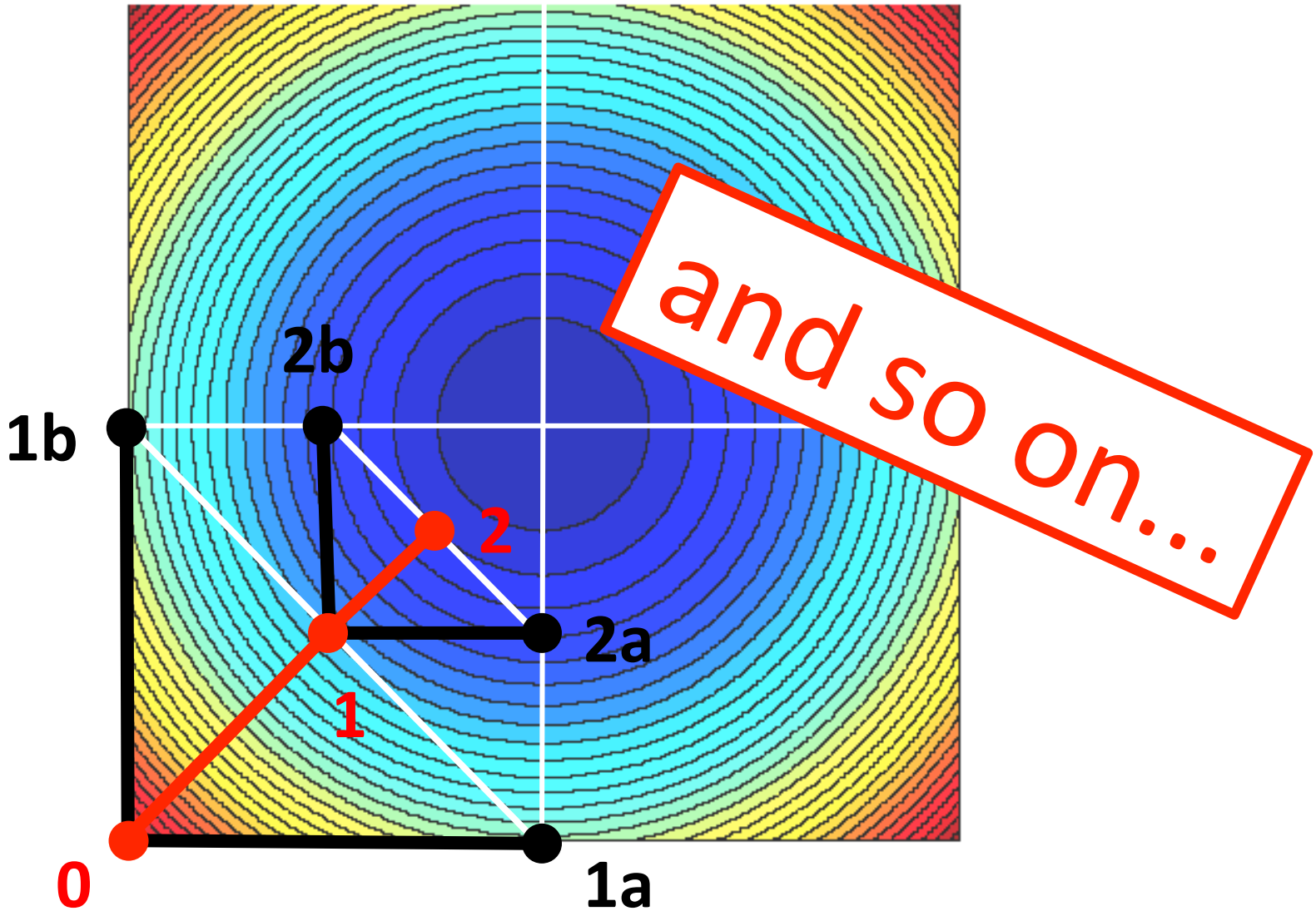
$$f(x^1, x^2) = (x^1 + x^2 - 1)^2$$

Idea: averaging updates may help



Averaging can be too conservative

$$f(x^1, x^2) = (x^1 - 1)^2 + (x^2 - 1)^2$$



Averaging may be too conservative 2

$$f(x) = (x^1 - 1)^2 + (x^2 - 1)^2 + \cdots + (x^n - 1)^2$$

But we wanted:

$$O\left(\frac{n\beta}{\tau} \times \xi(\epsilon)\right)$$

BAD!!!

$$k \geq \frac{n}{2} \log\left(\frac{n}{\epsilon}\right)$$

$$f(x_k) = n\left(1 - \frac{1}{n}\right)^{2k} \leq \epsilon$$

WANT

What to do?

Update to coordinate *i*

$$x_+ \leftarrow x + \frac{1}{\beta} \sum_{i=1}^{n} \boxed{h^i} e_i$$

i-th unit coordinate vector

Averaging: $\beta = n$

Summation: $\beta = 1$

Figure out when one can safely use: $\beta \approx 1$

Optimization Problems

Problem

Loss Regularizer

$$\min_{x \in \mathbb{R}^n} \left[\; F(x) \stackrel{\text{def}}{=} \boxed{f(x)} + \boxed{\Psi(x)} \; \right]$$

Convex
(smooth or nonsmooth)

Convex
(smooth or nonsmooth)
- separable
- allow $+\infty$

Regularizer: examples

$$\Psi(x) = \sum_{i=1}^{n} \Psi_i(x^i), \quad x = (x^1, x^2, \ldots, x^n)^T$$

e.g., LASSO

No regularizer $\Psi_i(x^i) \equiv 0$	Weighted L1 norm $\Psi_i(x^i) = \lambda_i \lvert x^i \rvert \qquad (\lambda_i > 0)$
Box constraints $\Psi_i(x^i) = \begin{cases} 0, & x^i \in X_i, \\ +\infty, & \text{otherwise.} \end{cases}$	Weighted L2 norm $\Psi_i(x^i) = \lambda_i (x^i)^2 \qquad (\lambda_i > 0)$

e.g., SVM dual

Loss: examples

	$f(x)$	
Quadratic loss	$\frac{1}{2}\|Ax-y\|_2^2 = \frac{1}{2}\sum_{j=1}^{m}(A_{j:}x-y_j)^2$	BKBG'11 RT'11b TBRS'13 RT '13a
Logistic loss	$\sum_{j=1}^{m}\log\left(1+\exp\left(-y_j A_{j:}x\right)\right)$	
Square hinge loss	$\frac{1}{2}\left(\max\left\{0, 1-y_j A_{j:}x\right\}\right)^2$	
L-infinity	$\|Ax-y\|_\infty = \max_{1\le j\le m}\|A_{j:}x-y_j\|$	FR'13
L1 regression	$\|Ax-b\|_1 = \sum_{j=1}^{m}\|A_{j:}x-y_j\|$	
Exponential loss	$\log\left(\frac{1}{m}\sum_{j=1}^{m}\exp\left(y_j A_{j:}x\right)\right)$	

3 models for f with small β

1 Smooth partially separable f [RT'11b AC/DC]

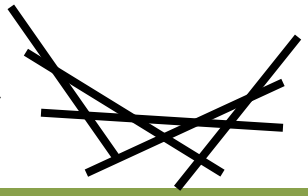
$$f(x+te_i) \le f(x) + (\nabla f(x))^T te_i + \frac{L_i}{2}t^2$$

$$f(x) = \sum_{J\in\mathcal{J}} f_J(x), \quad f_J \text{ depends on } x^i \text{ for } i \in J \text{ only}$$

$$\omega \stackrel{\text{def}}{=} \max_{J\in\mathcal{J}} |J|$$

2 Nonsmooth max-type f [FR'13]

$$f(x) = \max_{z\in Q} \{z^T Ax - g(z)\}$$



$$\omega \stackrel{\text{def}}{=} \max_{1\le j\le m} |\{i : A_{ji} \neq 0\}|$$

3 f with 'bounded Hessian' [BKBG'11, RT'13a HYDRA]

$$f(x+h) \le f(x) + (\nabla f(x))^T h + \tfrac{1}{2}h^T A^T A h$$

$$L = \mathrm{Diag}(A^T A)$$

$$\sigma \stackrel{\text{def}}{=} \lambda_{\max}(L^{-1/2} A^T A L^{-1/2})$$

General Theory

Randomized Parallel Coordinate Descent Method

New iterate

Current iterate

i-th unit coordinate vector

$$x_+ \leftarrow x + \frac{1}{\beta} \sum_{i \in \hat{S}} h^i e_i$$

Random set of coordinates (sampling)

Update to *i*-th coordinate

The update h^i depends on F, x and on the law describing $\hat{S}$

ESO: Expected Separable Overapproximation

$$x_+ \leftarrow x + \frac{1}{\beta} \sum_{i \in \hat{S}} \frac{1}{w_i} \nabla_i f(x) e_i$$

Minimize in *h*

$$\|h\|_w{}^2 \stackrel{\text{def}}{=} \sum_{i=1}^n w_i (h^i)^2$$

$$\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] \leq f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n} \boxed{\left((\nabla f(x))^T h + \frac{\beta}{2}\|h\|_w^2\right)}$$

$$h_{[\hat{S}]} = \sum_{i \in \hat{S}} h^i e_i$$

1. **Separable** in *h*
2. Can minimize in **parallel**
3. Can compute updates for $i \in \hat{S}$ **only**

Convergence rate: convex f

Theorem [RT'11b] If $(f, \hat{S}) \sim ESO(\beta, w)$, then

stepsize parameter

coordinates

$$k \geq \left(\frac{\beta n}{\boxed{\mathbf{E}[|\hat{S}|]}} \right) \left(\frac{2R_w^2(x_0, x_*)}{\epsilon} \right) \log \left(\frac{F(x_0) - F(x_*)}{\epsilon \rho} \right)$$

iterations

average # updated coordinates per iteration

error tolerance

implies

$$\mathbf{P}(F(x_k) - F(x_*) \leq \epsilon) \geq 1 - \rho$$

Convergence rate: strongly convex f

Theorem [RT'11b] If $(f, \hat{S}) \sim ESO(\beta, w)$, then

If $\mu_\psi(w)$ is large, then the slowdown effect of β is elliminated

Strong convexity constant of the regularizer Ψ

$$k \geq \left(\frac{n}{\mathbf{E}[|\hat{S}|]}\right)\left(\frac{\beta+\mu_\Psi(w)}{\mu_f(w)+\mu_\Psi(w)}\right)\log\left(\frac{F(x_0)-F(x_*)}{\epsilon\rho}\right)$$

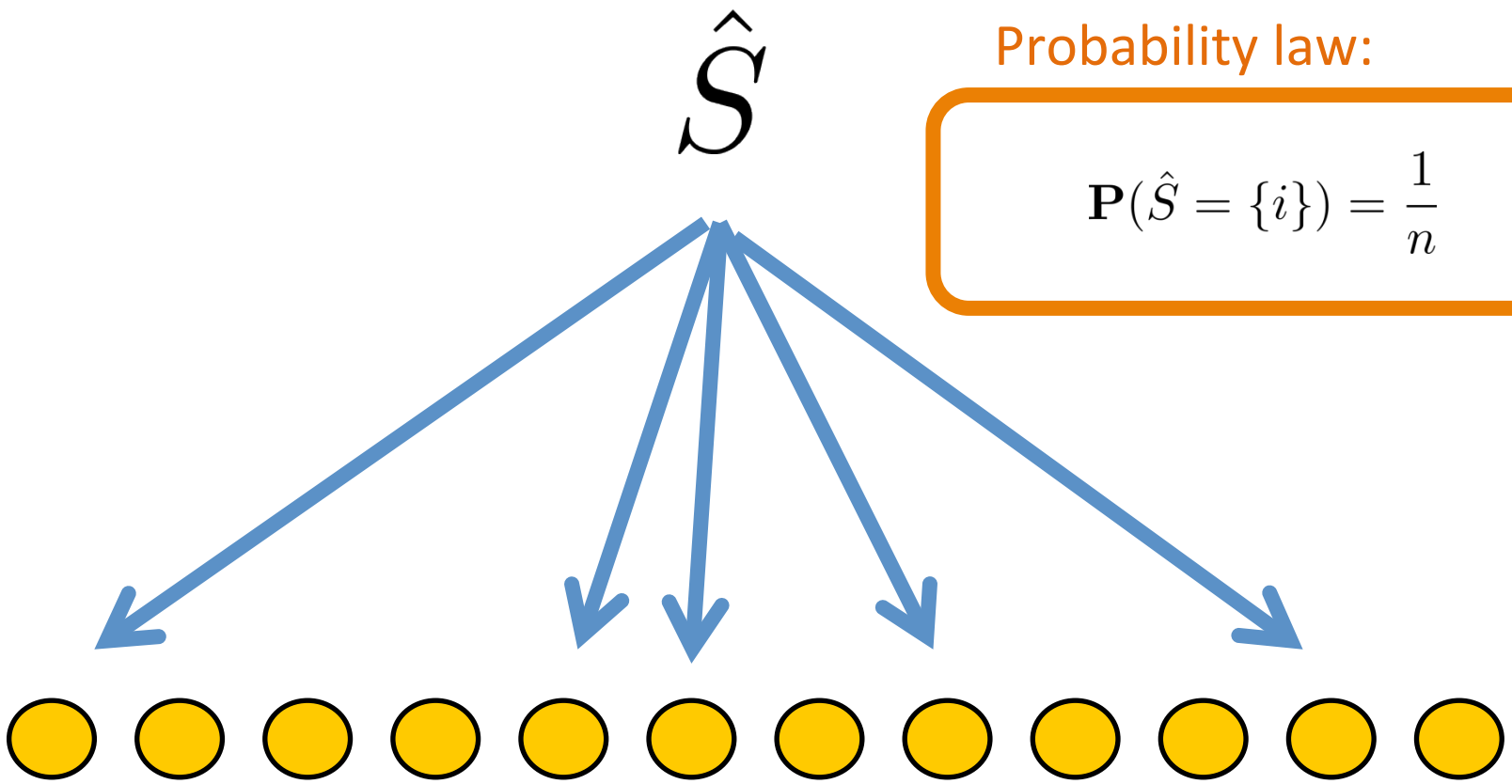
Strong convexity constant of the loss f

implies

$$\mathbf{P}(F(x_k) - F(x_*) \leq \epsilon) \geq 1 - \rho$$

Partial Separability and Doubly Uniform Samplings

Serial uniform sampling



Probability law:

$$\mathbf{P}(\hat{S} = \{i\}) = \frac{1}{n}$$

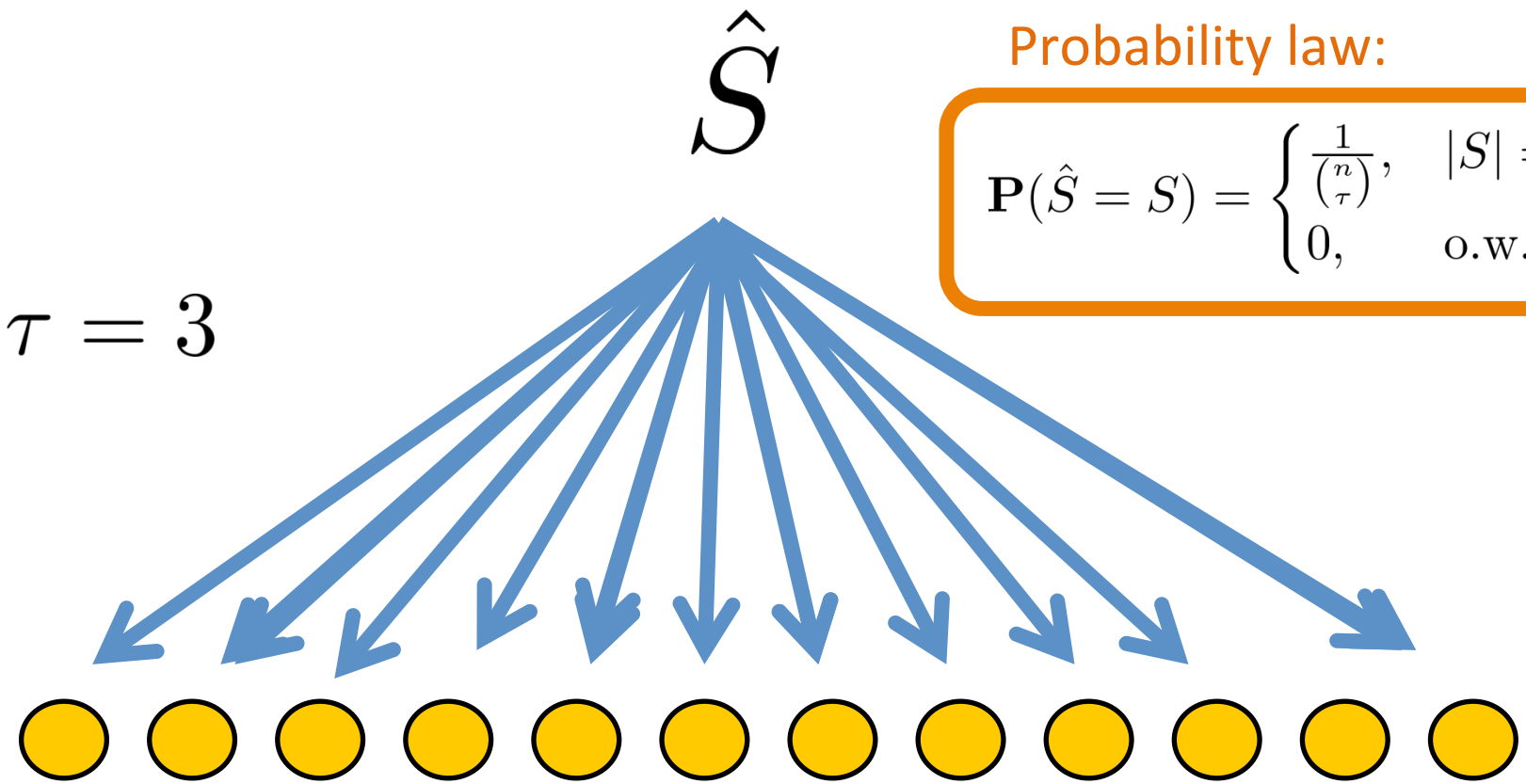
$n = 12 \qquad (\#\text{coordinates})$

τ-nice sampling

Good for shared memory systems

Probability law:

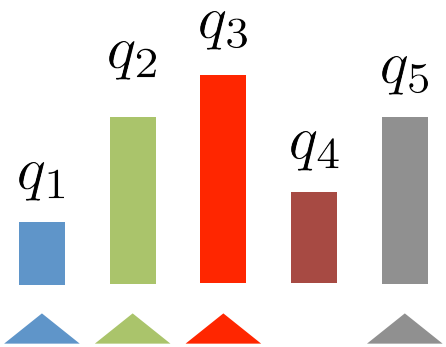
$$\mathbf{P}(\hat{S} = S) = \begin{cases} \frac{1}{\binom{n}{\tau}}, & |S| = \tau \\ 0, & \text{o.w.} \end{cases}$$



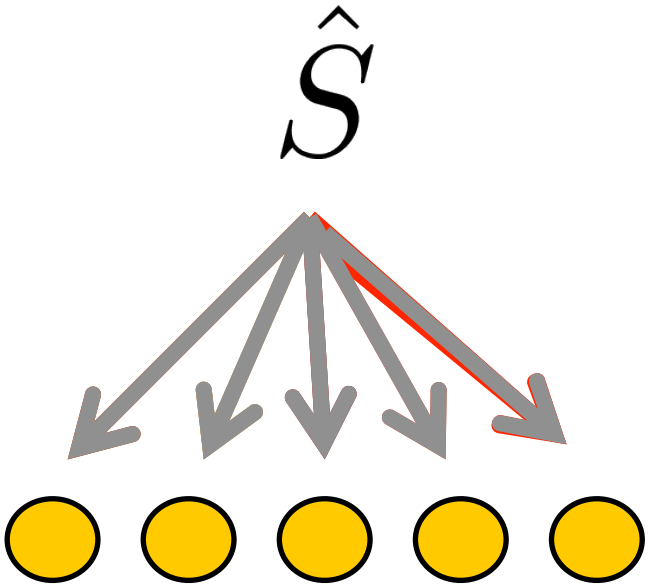
$n = 12 \qquad (\#\text{coordinates})$

Doubly uniform sampling

Can model unreliable processors / machines



$$q_\tau = \mathbf{P}(|\hat{S}| = \tau)$$



$n = 5 \qquad (\#\text{coordinates})$

Probability law:

$$\boxed{\mathbf{P}(\hat{S} = S) = \frac{q_{|S|}}{\binom{n}{|S|}}}$$

ESO for partially separable functions and doubly uniform samplings

1 Smooth partially separable f [RT'11b]

$$f(x+te_i) \le f(x) + (\nabla f(x))^T te_i + \frac{L_i}{2}t^2$$

$$f(x) = \sum_{J\in\mathcal{J}} f_J(x), \quad f_J \text{ depends on } x^i \text{ for } i \in J \text{ only}$$

$$\omega \stackrel{\text{def}}{=} \max_{J\in\mathcal{J}} |J|$$

Theorem [RT'11b]

If $\hat{S}$ is **doubly unifor**

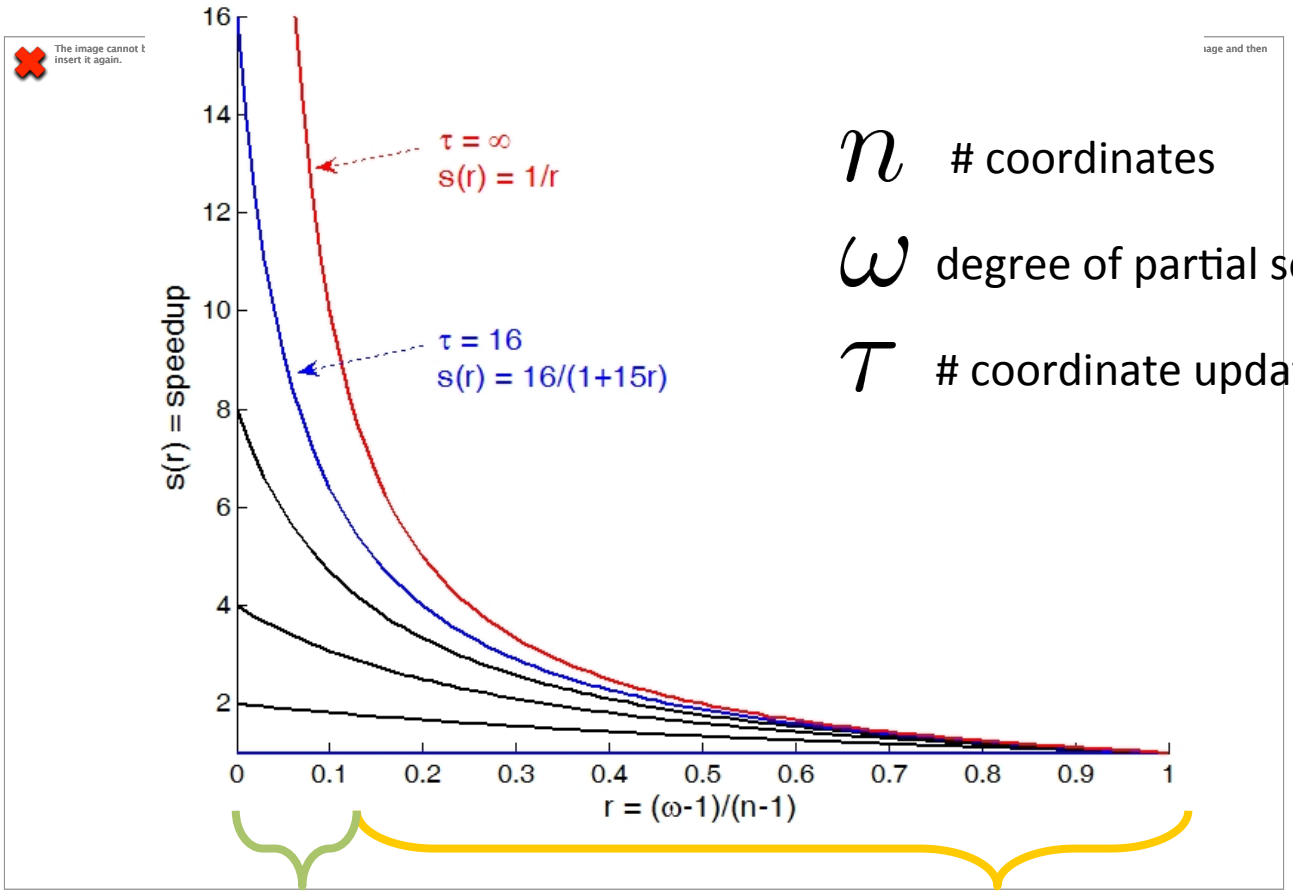
$$\mathbf{E}\left[f(x+h_{[\hat{S}]}) \quad \ldots \quad h + \frac{\beta}{2}\|h\|_w^2\right)$$

β is small if ω is small!
(i.e., more separable problems are easier)

$$\beta = 1 + \frac{(\omega-1)\left(\frac{\mathbf{E}[|\hat{S}|^2]}{\mathbf{E}[|\hat{S}|]} - 1\right)}{n-1}$$

$$w_i = L_i, \quad i = 1, 2, \ldots, n$$

Theoretical speedup



n # coordinates

ω degree of partial separability

τ # coordinate updates / iter

LINEAR OR GOOD SPEEDUP:
Nearly separable (sparse) problems

WEAK OR NO SPEEDUP:
Non-separable (dense) problems

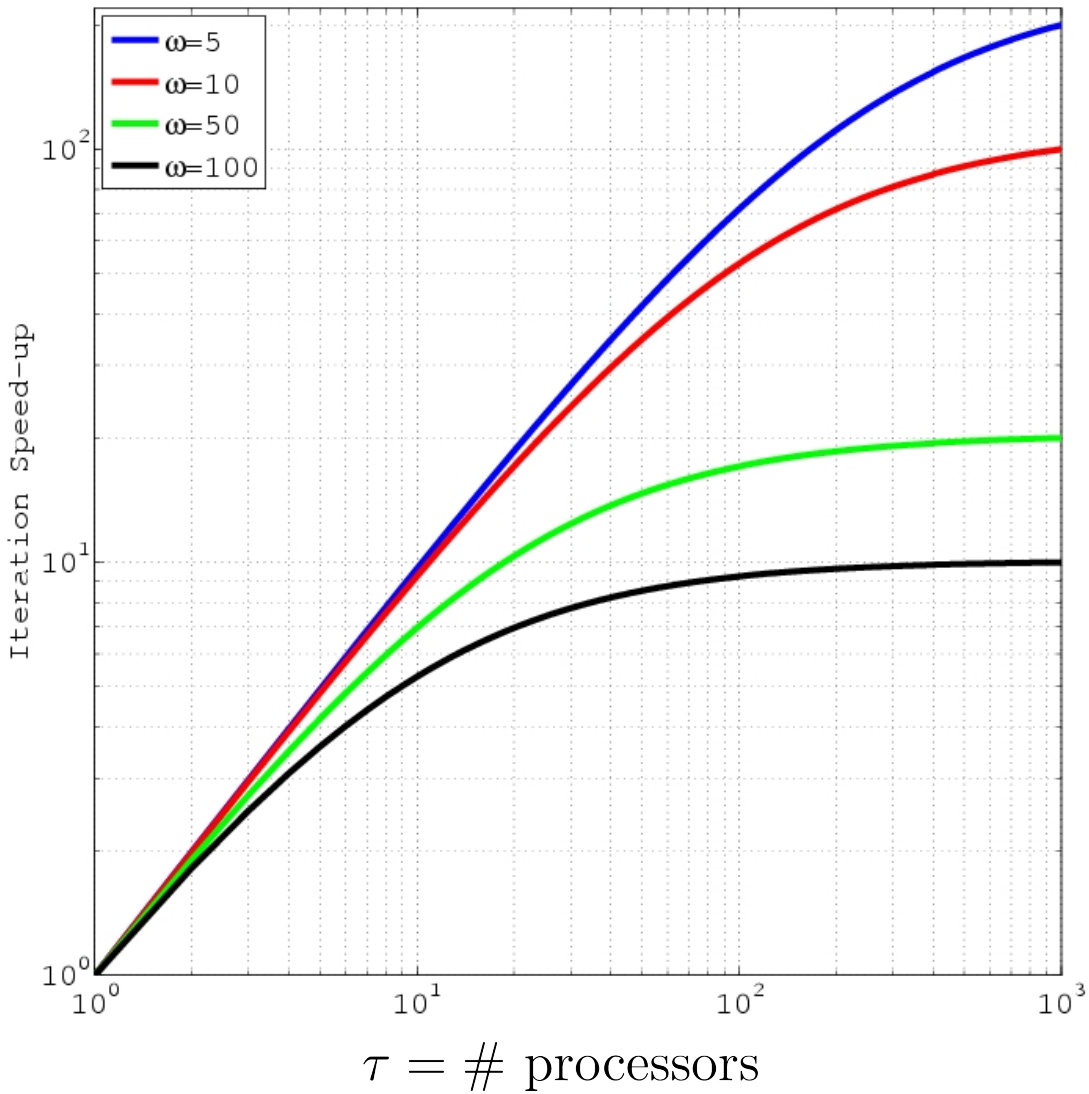
Much of Big Data is here!

Theory is when you know everything but nothing works.

Practice is when everything works but no one knows why.

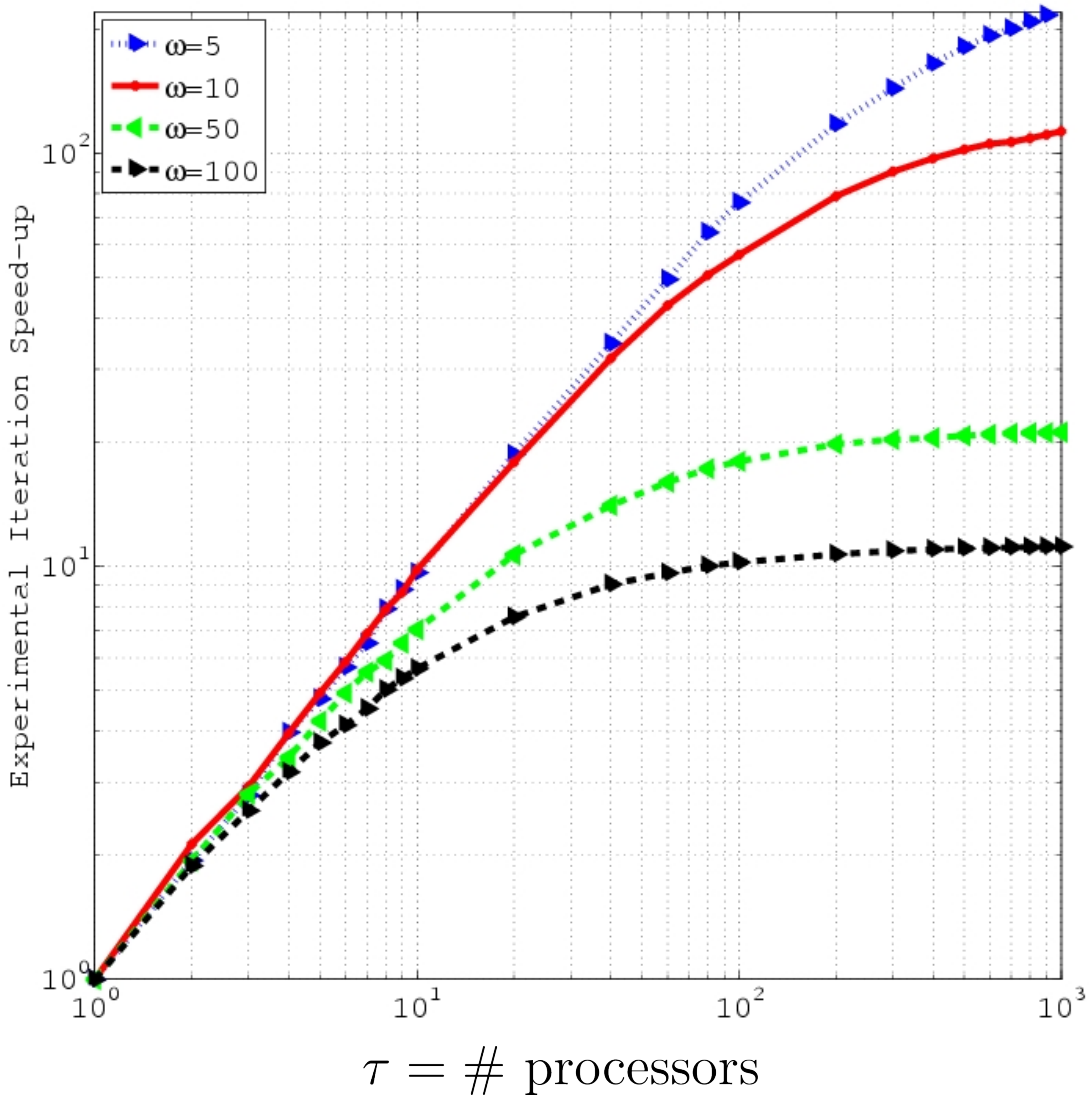
In our lab, theory and practice are combined: nothing works and no one knows why.

Theory



n = 1000
(# coordinates)

Practice



n = 1000
(# coordinates)

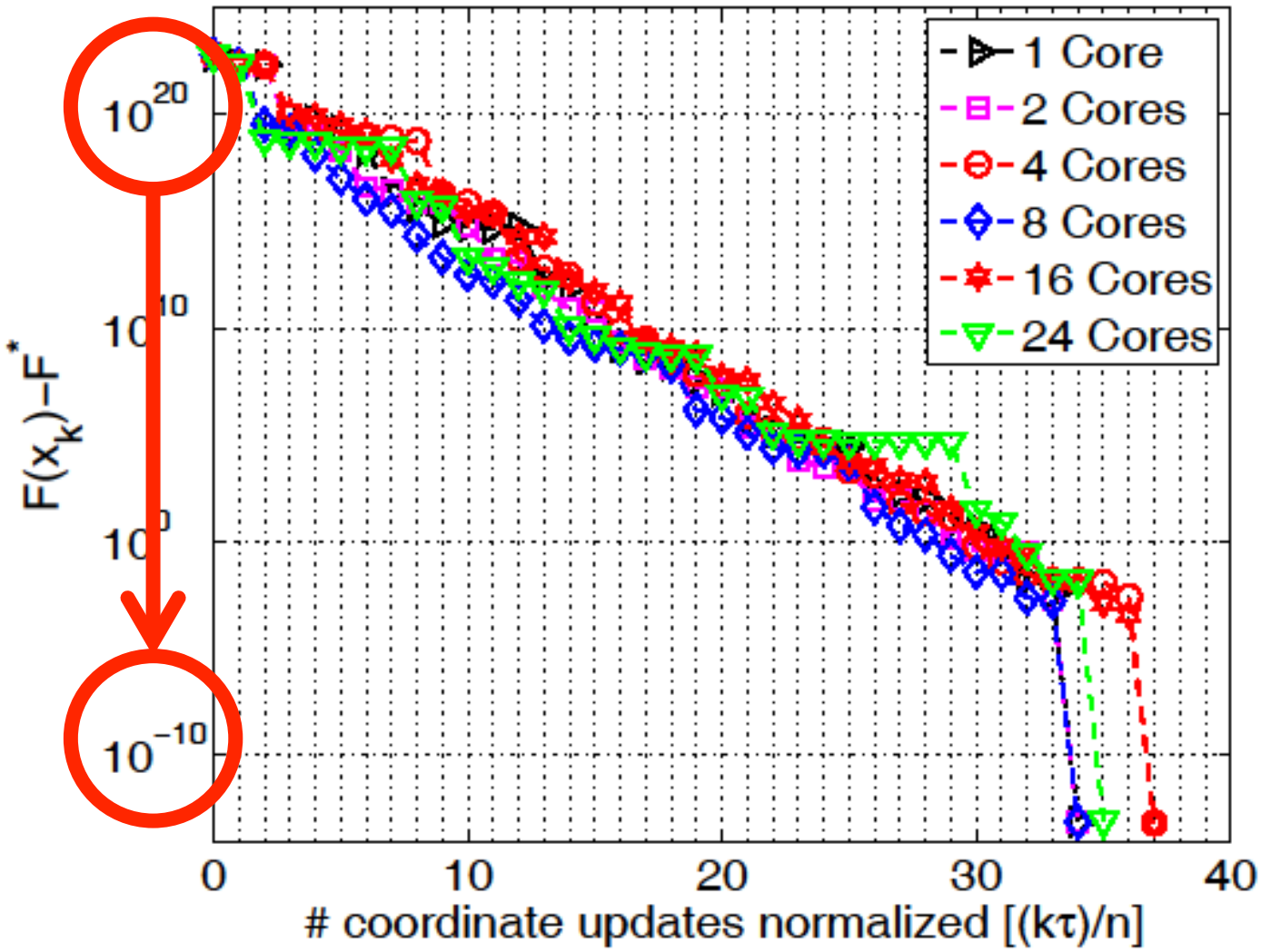
Experiment with a 1 billion-by-2 billion LASSO problem

Optimization with Big Data

= Extreme* Mountain Climbing

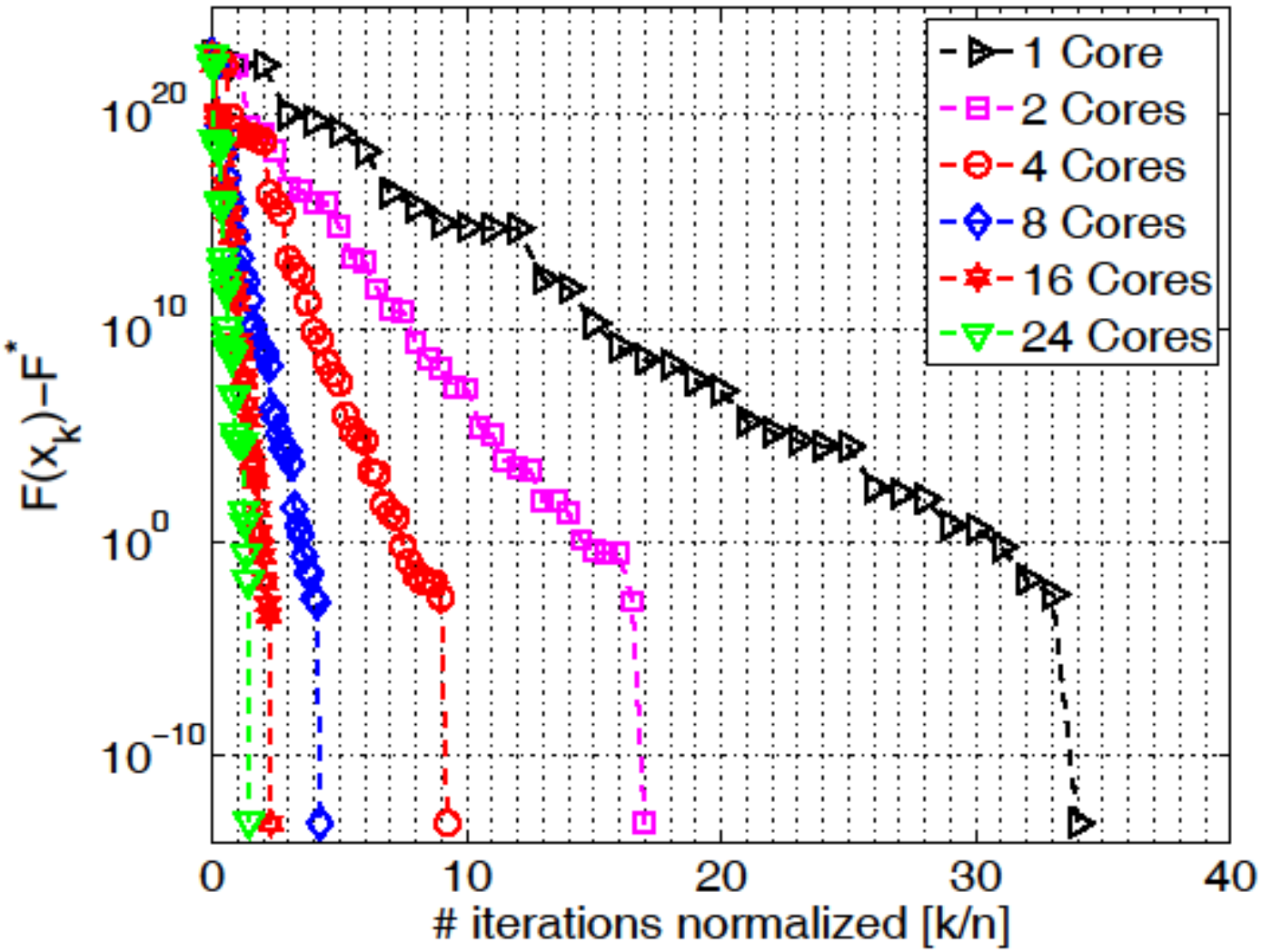
* in a billion dimensional space on a foggy day

Coordinate Updates



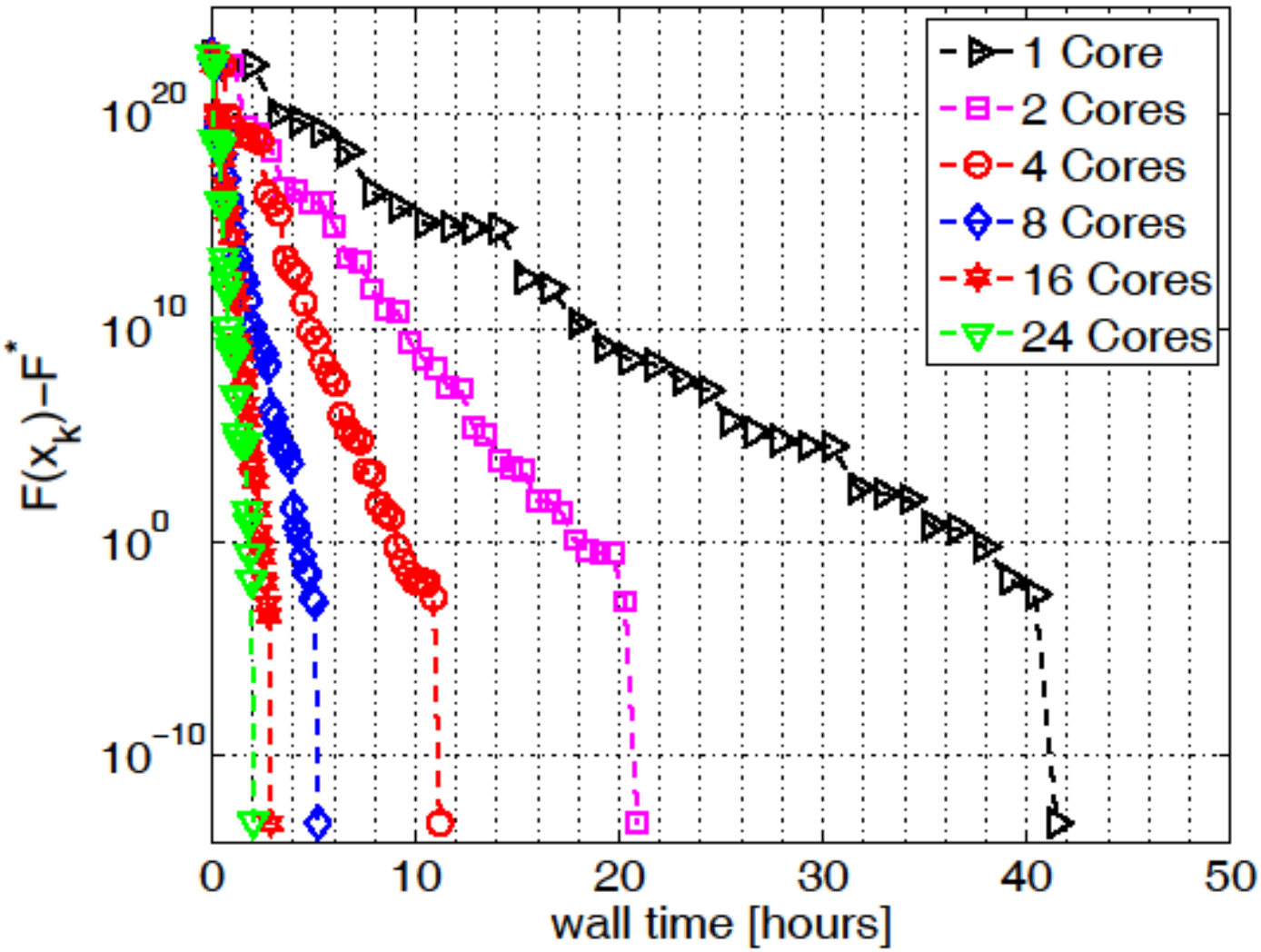
LASSO problem with $A \in \mathbb{R}^{m \times n}$, where $n = 10^9$ and $m = 2 \times 10^9$

Iterations



LASSO problem with $A \in \mathbb{R}^{m \times n}$, where $n = 10^9$ and $m = 2 \times 10^9$

Wall Time



LASSO problem with $A \in \mathbb{R}^{m \times n}$, where $n = 10^9$ and $m = 2 \times 10^9$

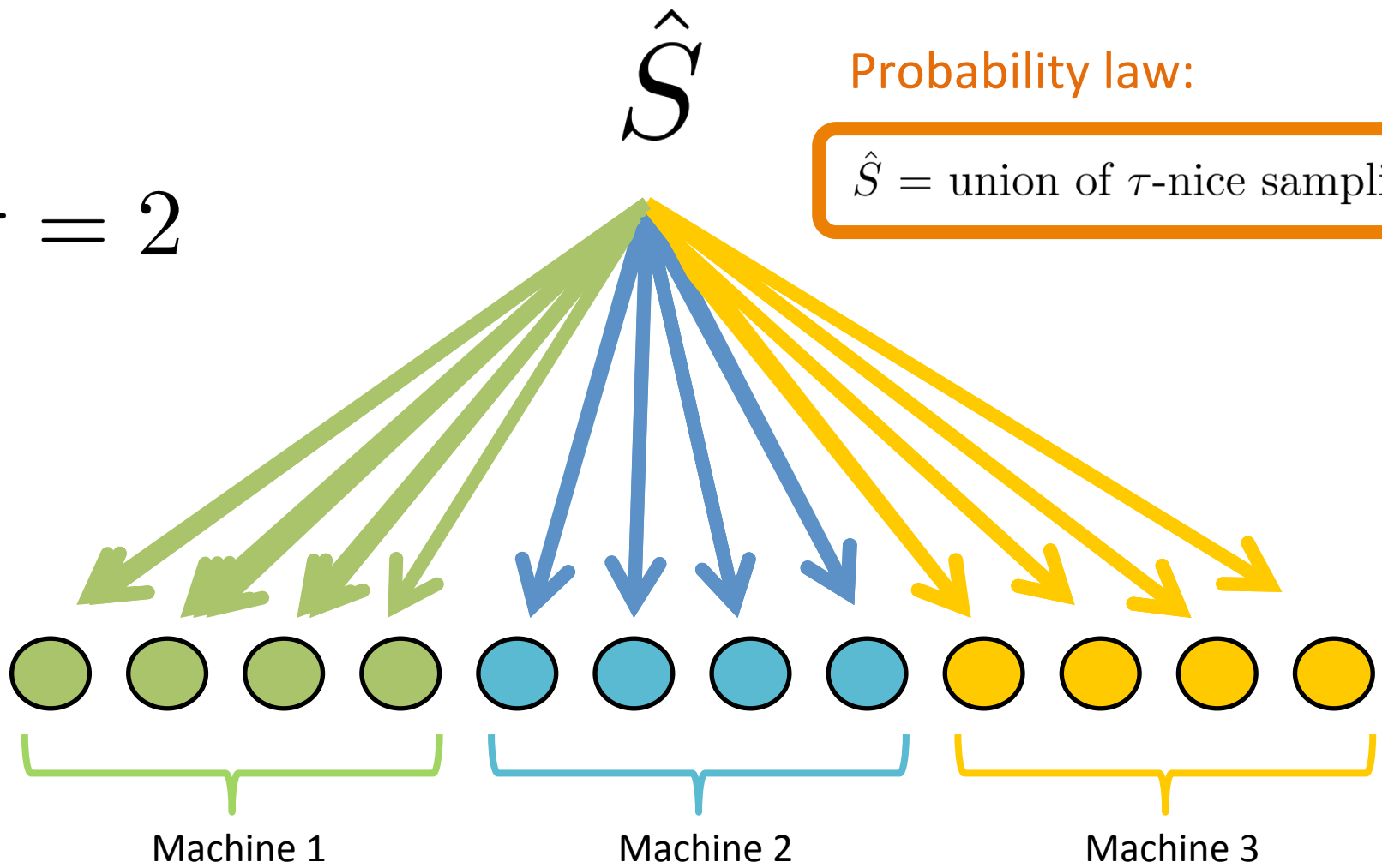
Distributed-Memory Coordinate Descent

HYDRA

Distributed τ-nice sampling

Good for a distributed version of coordinate descent

$$\tau = 2$$



Probability law:

$$\boxed{\hat{S} = \text{union of } \tau\text{-nice samplings}}$$

ESO: Distributed setting

3 *f* with 'bounded Hessian' [BKBG'11, RT'13a HYDRA]

$$f(x+h) \le f(x) + (\nabla f(x))^T h + \tfrac{1}{2} h^T A^T A h$$

$$L = \mathrm{Diag}(A^T A)$$

$$\sigma \overset{\text{def}}{=} \lambda_{\max}(L^{-1/2} A^T A L^{-1/2})$$

spectral norm of the **data**

Theorem [RT'13b]

If $\hat{S}$ is **distributed τ-nice sampling** (with c nodes, each owning $s \overset{\text{def}}{=} n/c$ coordinates) and Assumption 1 holds, then

$$\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] \le f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n}\left((\nabla f(x))^T h + \frac{\beta}{2}\|h\|_w^2\right)$$

$$\beta = 1 + \frac{(\sigma-1)(\tau-1)}{n-1} + \left(\frac{\tau}{s} - \frac{\tau-1}{s-1}\right)\frac{\sigma'-1}{\sigma'}\sigma$$

$$w_i = L_{ii}, \quad i = 1, 2, \ldots, n$$

Bad partitioning at most doubles # of iterations

spectral norm of the **partitioning**

$$\beta = \underbrace{1 + \frac{(\sigma-1)(\tau-1)}{n-1}}_{\beta_1} + \underbrace{\left(\frac{\tau}{s} - \frac{\tau-1}{s-1}\right)\frac{\sigma'-1}{\sigma'}\sigma}_{\beta_2}$$

Theorem [RT'13b] If $\tau \geq 2$, then $\beta_2 \leq \beta_1$

implies

iterations = $O\left(\frac{2\beta_1 n}{\tau c}\right)$

updates/node

nodes

Distributed Coordinate Descent for Big Data Optimization

Martin Takáč Jakub Mareček Peter Richtárik
University of Edinburgh

NAIS Centre for Numerical Algorithms and Intelligent Software

1. Problem Formulation

$$\min_{x\in\mathbb{R}^n}[F(x) \equiv f(x) + \Psi(x)]$$

1. f convex, partially separable of degree ω and $\forall x \in \mathbb{R}^n, t \in \mathbb{R}$ and $i \in \{1,2,\ldots,n\}$ satisfying $|\nabla_i f(x) - \nabla_i f(x + te_i)| \le L_i|t|$, where L_i are coordinate Lipschitz constants
2. Ψ convex and separable ($\Psi(x) = \sum_i \Psi_i(x^i)$)
3. Description of f so large that it does not fit onto a single computer! $\Rightarrow$ a cluster of C nodes

2. The Algorithm

Pre-processing: Partition coordinates $\{1,2,\ldots,n\}$ to C sets $S_1, S_2, \ldots, S_C$

In one iteration computers $c = 1,2,\ldots,C$ in parallel do

1. Choose random $\hat{S}_c \subset S_c$
2. For each $i \in \hat{S}_c$ in parallel compute $t_i^* \leftarrow \arg\min_{t\in\mathbb{R}} \nabla_i f(x_k)t + \beta\frac{L_i}{2}t^2 + \Psi_i(x_k^i + t)$
3. $x_{k+1} \leftarrow x_k + \sum_{i\in\hat{S}_c} t_i^* e_i$

2. Distributed Sampling

We can analyze the above algorithm under the following assumptions:

- $|S_c| = \frac{n}{C}$ for all $c = 1,2,\ldots,C$
- $\hat{S}_c$ is chosen uniformly as one of the subsets of S_c of cardinality τ

Distributed sampling: $\hat{S} = \cup_{c=1}^{C}\hat{S}_c$

$$\Downarrow$$

$$\beta := 1 + \frac{-C(\tau-1) + \omega[\tau C(1 + \frac{C-1}{n}) - 1]}{\max\{n-C, 1\}} \quad \text{(see [1])}$$

Special cases:

- $C = 1 \Rightarrow \beta = 1 + \frac{(\tau-1)(\omega-1)}{\max\{n-1,1\}}$ (see [2])
- $C = \tau = 1 \Rightarrow \beta = 1$ (see [3])

However, we need new analysis for the $C > 1$ case.

4. Complexity Theorem

$$k \ge \frac{\beta n}{\tau C}\frac{2R^2}{\epsilon}\log\left(\frac{F(x_0) - F^*}{\epsilon\rho}\right)$$

$$\Downarrow$$

$$\text{Prob}(F(x_k) - F(x_*) \le \epsilon) \ge 1 - \rho$$

$$(R^2 \approx \sum_i L_i(x_0^i - x_*^i)^2)$$

5. AC/DC Solver

We developed a solver (http://code.google.com/p/ac-dc/) for

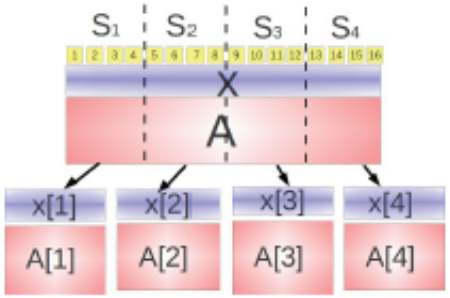
$$f(x) = \sum_{i=1}^{m} Loss(x; A_j, b_j), \qquad \Psi(x) = \lambda\|x\|_1$$

3 supported losses	$Loss(x, A_j, b_j)$
square loss	$\frac{1}{2}(b_j - A_j x)^2$
logistic loss	$\log(1 + e^{-b_j A_j x})$
hinge square loss	$\frac{1}{2}\max\{0, 1 - b_j A_j x\}^2$

Note that $A_j \in \mathbb{R}^n$ is a row vector and later will represent the j-th row of matrix A.

6. Data Distribution

Assume that we have $C = 4$ compute nodes and $n = 16$ coordinates. The coordinates can be partitioned into 4 balanced groups $\{S_1, S_2, S_3, S_4\}$.



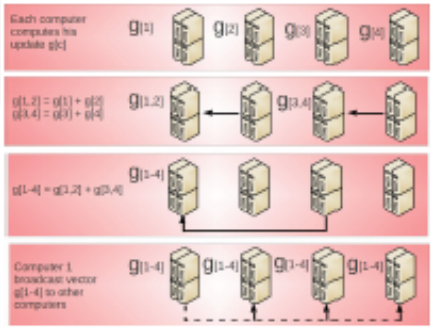
On computer 1, only the first 4 coordinates of vector x are stored and also the corresponding 4 columns of matrix A. Data distribution is crucial for problems whose size exceeds available memory of a single computer!

7. Implementation Details (square loss example)

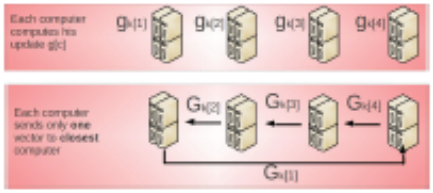
If we can maintain $g_k = Ax_k - b$ on all computers, then since $\nabla_i f(x_k) = \langle a_i, g_k\rangle$ (a_i is the i-th column of matrix A), computer c can compute $\nabla_i f(x_k)$ for $i \in S_c$, and hence the algorithm can be run.

- Note that $g_{k+1} = Ax_{k+1} - b = A(x_k + \sum_{c=1}^{C}\sum_{i\in\hat{S}_c} t_i^* e_i) - b = g_k + \sum_{c=1}^{C}\sum_{i\in\hat{S}_c} a_i t_i^*$
- That is, computer c additively contributes $g_k[c] := \sum_{i\in\hat{S}_c} a_i t_i^*$ to the update of g_k
- So, we need to add up the distributed updates $g_k[c]$

Reduce All (RA)

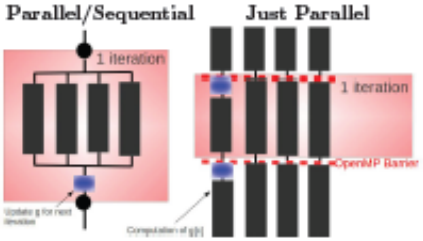


Asynchronous StreamLine (ASL)



- $G_k[c] = G_{k-1}[Prev(c)] + g_k[c] - g_{k-C}[c]$
- $g_{k+1}^c = g_k^c + g_k[c] + G_k[Prev(c)] - g_{k-C}[c]$
- ASL: much LESS communication that RA!
- ASL: asynchronous (non-blocking) communication
- ASL: communication only between two closest computers

8. Hybrid Implementations

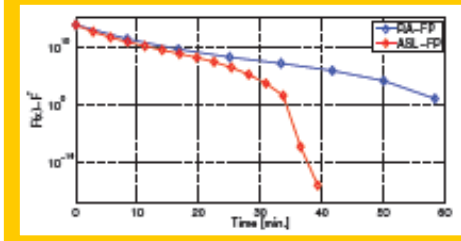


Left image shows Parallel and Serial (PS) approach, where each MPI process runs few OpenMP threads for computing t_i^* and $g_k[c]$ (black boxes) and afterwards, MPI communication takes places (blue boxes). Right image shows Fully Parallel (FP) approach in which one of the threads deals with communication and when waiting for a new communication, it helps the other threads to do some computation.

9. Numerical Experiments

All experiments were done on HECToR - Cray XE6 using 2,048 cores. Problem size $A \in \mathbb{R}^{10^6\times 8\cdot 10^9}$ had 1.2 TBytes and we used $\tau = 10^3$.

method	avg. time / iter.
RA-PS	2.252
RA-FP	2.052
ASL-FP	0.691

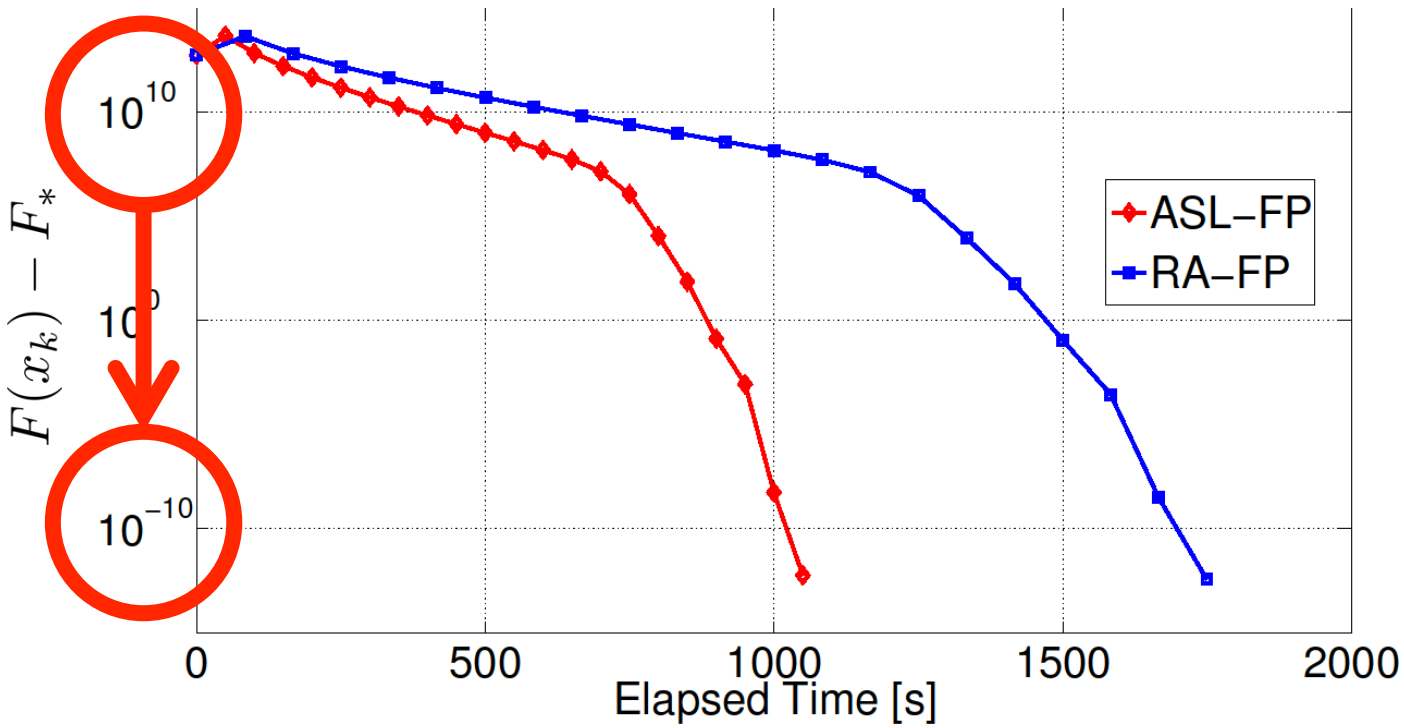


10. References

[1] Takáč, M., Mareček, J. and Richtárik, P.: Distributed coordinate descent methods for big data optimization, 2013
[2] Richtárik, P., Takáč, M.: Parallel coordinate descent methods for big data optimization, 2012
[3] Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function, Mathematical Programming, 2012

LASSO with a 3TB data matrix

$$A = 10^9\text{-by-}\underbrace{0.5 \times 10^9}_{n \;=\; \#\text{ coordinates}}$$



128 Cray XE6 nodes with 4 MPI processes (c = 512)

Each node: 2 x 16-cores with 32GB RAM

Conclusions

- Coordinate descent methods scale very well to big data problems of special structure
  - Partial separability (sparsity)
  - Small spectral norm of the data
  - Nesterov separability, ...
- Care is needed when combining updates (add them up? average?)

References: serial coordinate descent

- Shai Shalev-Shwartz and Ambuj Tewari, **Stochastic methods for L1-regularized loss minimization.** *JMLR 2011.*

- Yurii Nesterov, **Efficiency of coordinate descent methods on huge-scale optimization problems.** *SIAM Journal on Optimization, 22(2):341-362, 2012.*

- [RT'11b] P.R. and Martin Takáč, **Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function**. *Mathematical Prog.*, 2012.

- Rachael Tappenden, P.R. and Jacek Gondzio, **Inexact coordinate descent: complexity and preconditioning**, *arXiv: 1304.5530, 2013.*

- Ion Necoara, Yurii Nesterov, and Francois Glineur. **Efficiency of randomized coordinate descent methods on optimization problems with linearly coupled constraints.** *Technical report, Politehnica University of Bucharest, 2012.*

- Zhaosong Lu and Lin Xiao. **On the complexity analysis of randomized block-coordinate descent methods.** *Technical report, Microsoft Research, 2013.*

References: parallel coordinate descent

- [BKBG'11] Joseph Bradley, Aapo Kyrola, Danny Bickson and Carlos Guestrin, **Parallel Coordinate Descent for L1-Regularized Loss Minimization**. *ICML 2011*
- [RT'12] P.R. and Martin Takáč, **Parallel coordinate descen methods for big data optimization**. *arXiv:1212.0873,* 2012
- Martin Takáč, Avleen Bijral, P.R., and Nathan Srebro. **Mini-batch primal and dual methods for SVMs.** *ICML 2013*
- [FR'12] Olivier Fercoq and P.R., **Smooth minimization of nonsmooth functions with parallel coordinate descent methods**. *arXiv:1309.5885,* 2013
- [RT'13a] P.R. and Martin Takáč, **Distributed coordinate descent method for big data learning**. *arXiv:1310.2059,* 2013
- [RT'13b] P.R. and Martin Takáč, **On optimal probabilities in stochastic coordinate descent methods**. *arXiv:1310.3438,* 2013

Good entry point to the topic (4p paper)

References: parallel coordinate descent

- P.R. and Martin Takáč, **Efficient serial and parallel coordinate descent methods for huge-scale truss topology design.** *Operations Research Proceedings 2012.*

- Rachael Tappenden, P.R. and Burak Buke, **Separable approximations and decomposition methods for the augmented Lagrangian.** *arXiv:1308.6774, 2013.*

- Indranil Palit and Chandan K. Reddy. **Scalable and parallel boosting with MapReduce.** *IEEE Transactions on Knowledge and Data Engineering, 24(10): 1904-1916, 2012.*

- Shai Shalev-Shwartz and Tong Zhang, **Accelerated mini-batch stochastic dual coordinate ascent.** *NIPS 2013.*

TALK TOMORROW