Reducing Complexity in Algebraic Multigrid

Hans De Sterck

Department of Applied Mathematics

University of Waterloo

Ulrike Yang

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

University of
Waterloo

CASC

Outline

- **introduction: AMG**

- **complexity growth when using classical coarsenings**

- **Parallel Modified Independent Set (PMIS) coarsening**

- **scaling results**

- **improving interpolation**

- **conclusions and future work**

Introduction

- *solve* $\mathbf{Au = f}$

- $\mathbf{A}$ *from 3D PDE – sparse!*

- *large problems (10^9 dof) - parallel*

- *unstructured grid problems*

Algebraic Multigrid (AMG)
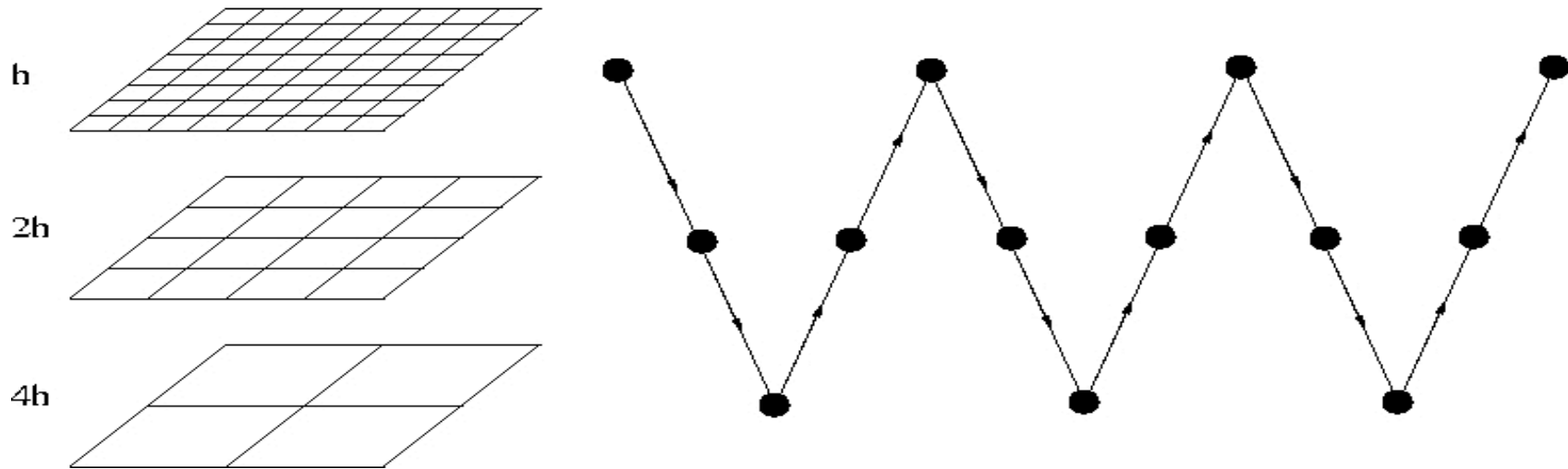
- *multi-level*
- *iterative*
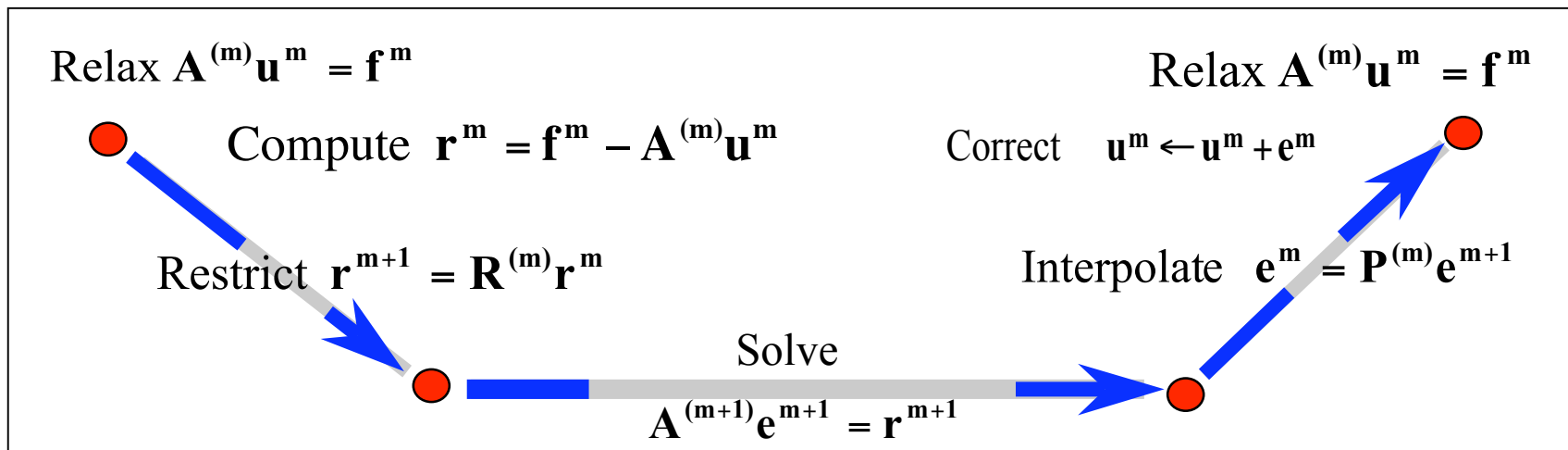- *algebraic: suitable for unstructured!*

AMG building blocks

Setup Phase:

- — Select coarse "grids"
- — Define interpolation, $P^{(m)}$, $m = 1,2,...$
- — Define restriction and coarse-grid operators

$$R^{(m)} = P^{(m)T} \qquad A^{(m+1)} = P^{(m)T} A^{(m)} P^{(m)}$$

Solve Phase

Relax $A^{(m)} u^m = f^m$

Compute $r^m = f^m - A^{(m)} u^m$

Restrict $r^{m+1} = R^{(m)} r^m$

Solve

$$A^{(m+1)} e^{m+1} = r^{m+1}$$

Relax $A^{(m)} u^m = f^m$

Correct $u^m \leftarrow u^m + e^m$

Interpolate $e^m = P^{(m)} e^{m+1}$

AMG complexity - scalability

- *Operator complexity* $C_{op} = \dfrac{\sum_i \text{nonzeros}(A_i)}{\text{nonzeros}(A_0)}$

e.g., 3D, ideally: $C_{op} = 1 + 1/8 + 1/64 + \ldots < 8/7$

measure of *memory use*, and *work in solve phase*

- scalable algorithm:

  O(n) operations per V-cycle (C_{op} bounded)

  AND

  number of V-cycles independent of n
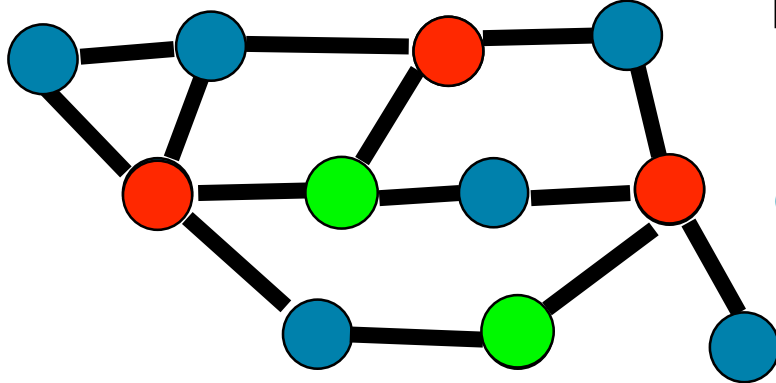
  (ρ_{AMG} independent of n)

AMG coarsening and interpolation

- *large a_{ij}, 'strong connections' are important*

- *define strength matrix S:*

$$A = \begin{bmatrix} x & x & x & & \\ & x & x & & x \\ & x & x & x & x \\ & x & x & x & \\ x & & & x & x \end{bmatrix} \qquad S = \begin{bmatrix} 1 & 1 & 0 & & \\ & 1 & 0 & & 1 \\ 0 & 0 & 1 & 1 \\ & 1 & 0 & 1 & \\ 1 & & & 1 & 0 \end{bmatrix}$$

- *consider the undirected graph of S*

- *apply parallel maximal independent set algorithm to graph(S) [Luby, 1986]*

classical AMG coarsening (CLJP)

- **(C1) Maximal Independent Set:**

Independent: no two **C**-points are connected

Maximal: if one more **C**-point is added, the independence is lost

- **(C2) All F-F connections require connections to a common C-point (for good interpolation)**

- **F-points have to be changed into C-points, to ensure (C2); (C1) is violated**

more C-points, higher complexity

Classical coarsening: scalability results

- *example: finite difference Laplacian, parallel CLJP coarsening algorithm*

- *2D (5-point): near-optimal scalability* **(250^2 dof/proc)**

Procs	C_{op}	t_{tot}	Iter
16	4.48	2.89	9
64	4.50	3.85	9
256	4.50	5.01	9

Classical coarsening: complexity growth in some cases

- *3D (7-point): complexity growth*

dof	C_{op}
32^3	16.17
64^3	22.51

Classical coarsening: complexity growth in some cases

- *4D (9-point), 5D (11-point): complexity growth!!*

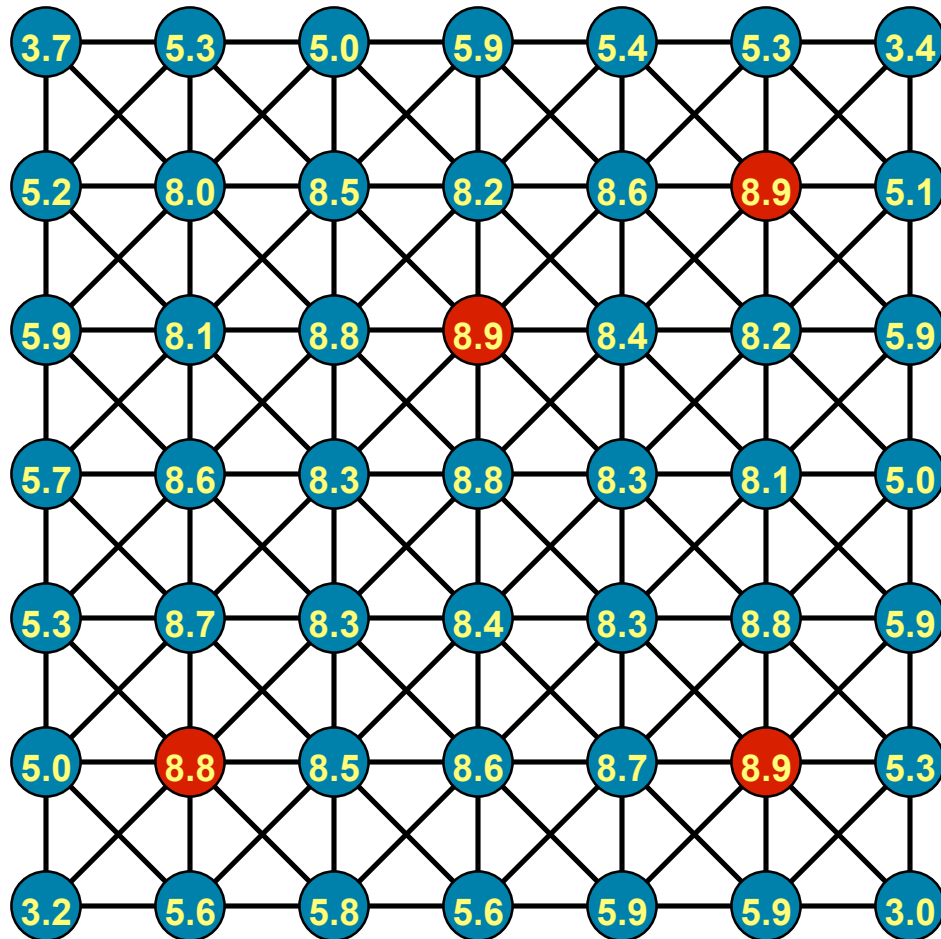|	dof	C_{op}
4D	20^4	127.5
5D	9^5	256.9

- *excessive memory use*

(results obtained by Jeff Butler)

our approach to reduce complexity

- *do not add C points for strong F-F connections that do not have a common C point*

- *less C points, reduced complexity, but worse convergence factors expected*

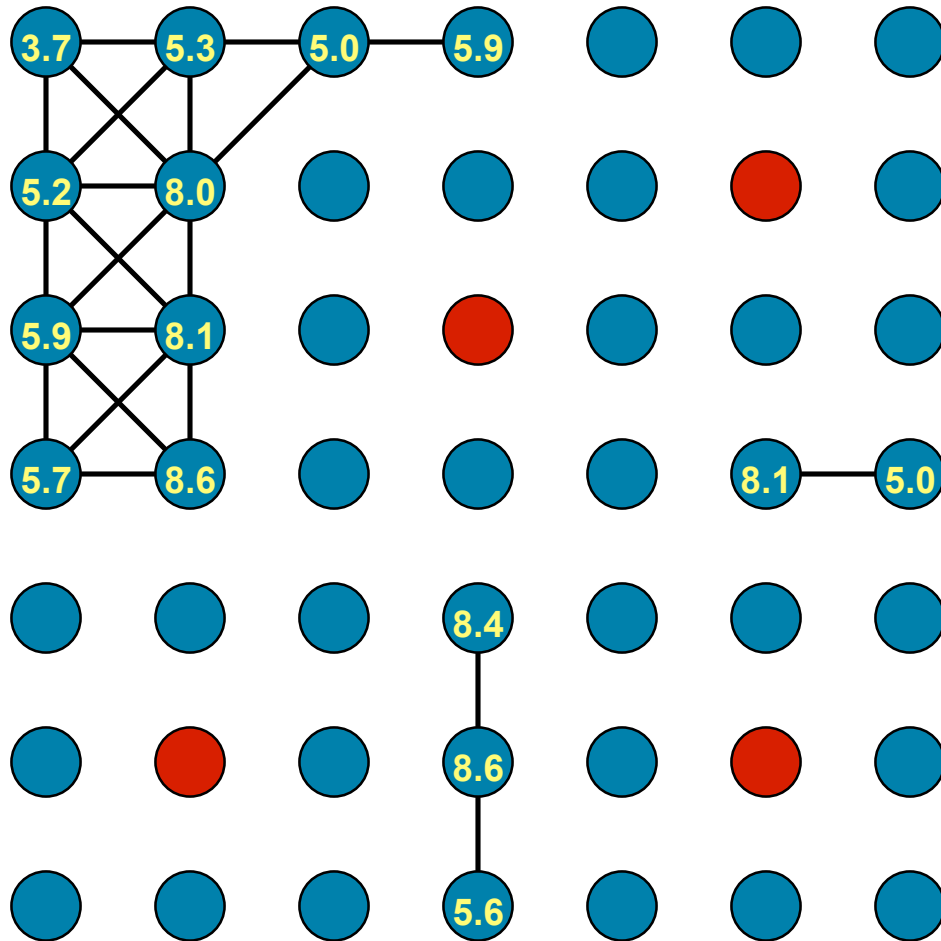- *compensate by GMRES acceleration*

PMIS: select 1

➡ **select C-pts with maximal measure locally**

➡ **make neighbour F-pts**

➡ **remove neighbour edges**

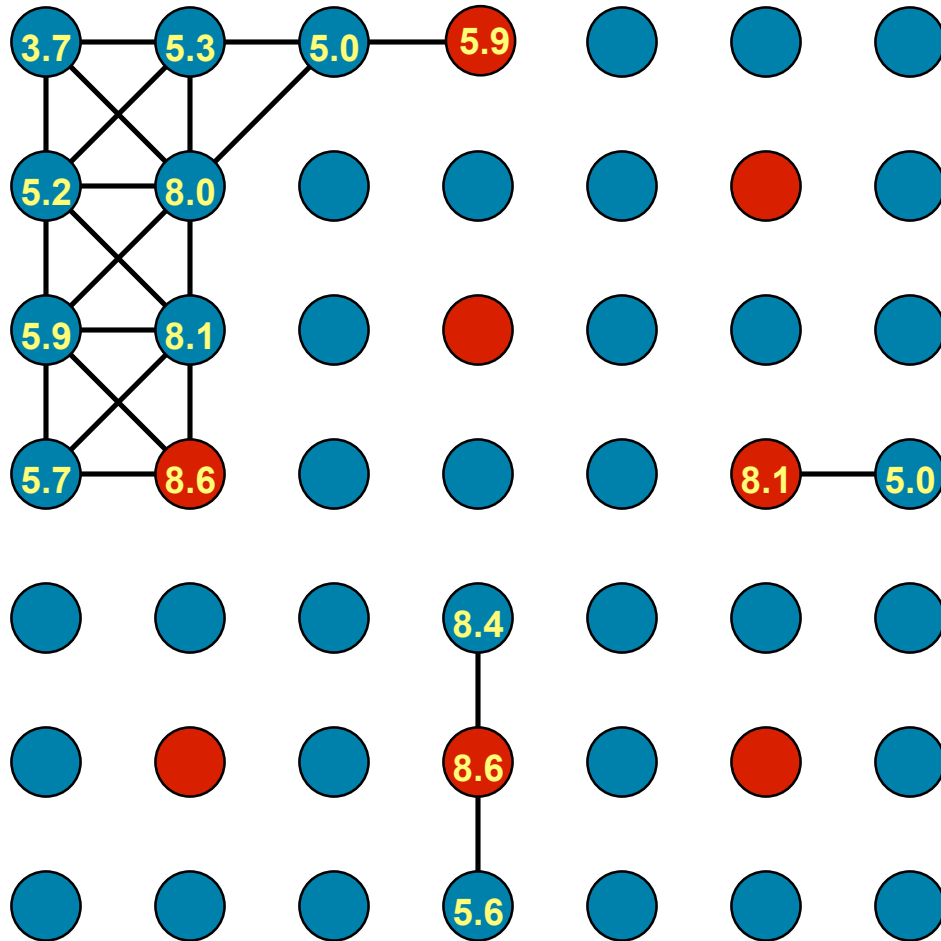# PMIS: remove and update 1

→ **select C-pts with maximal measure locally**

→ **make neighbours F-pts**

→ **remove neighbour edges**

PMIS: select 2
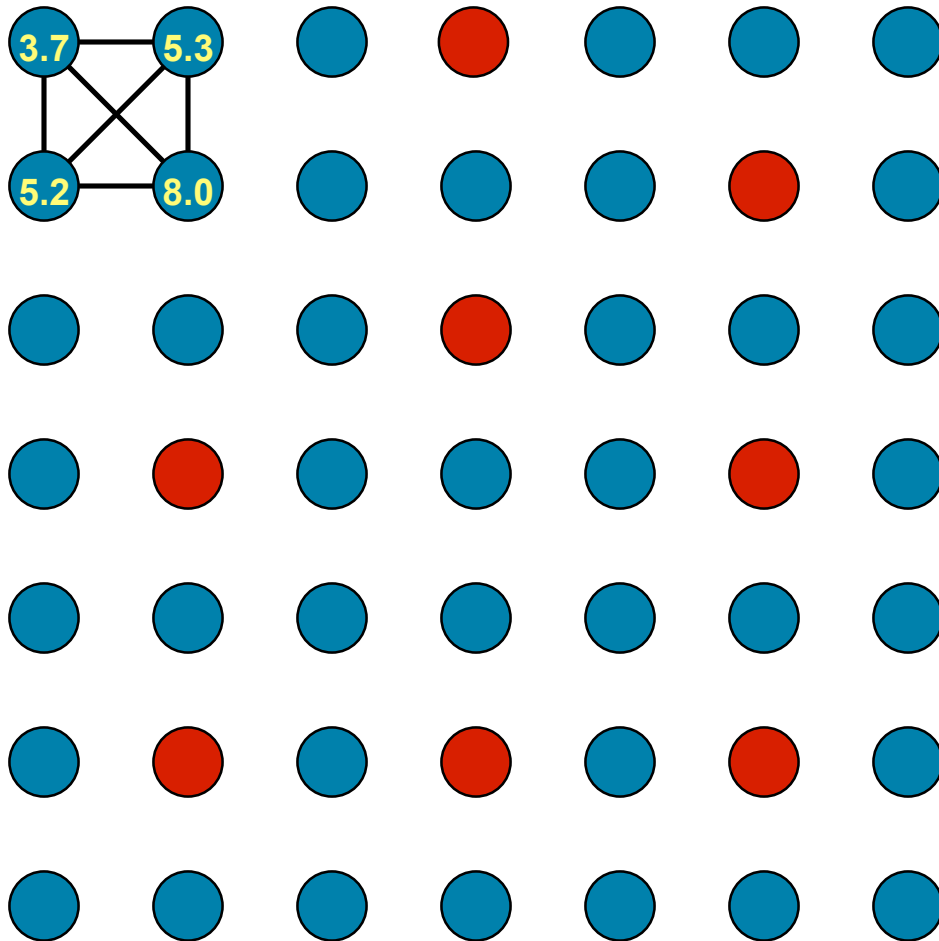

➡ **select C-pts with maximal measure locally**

➡ make neighbours F-pts

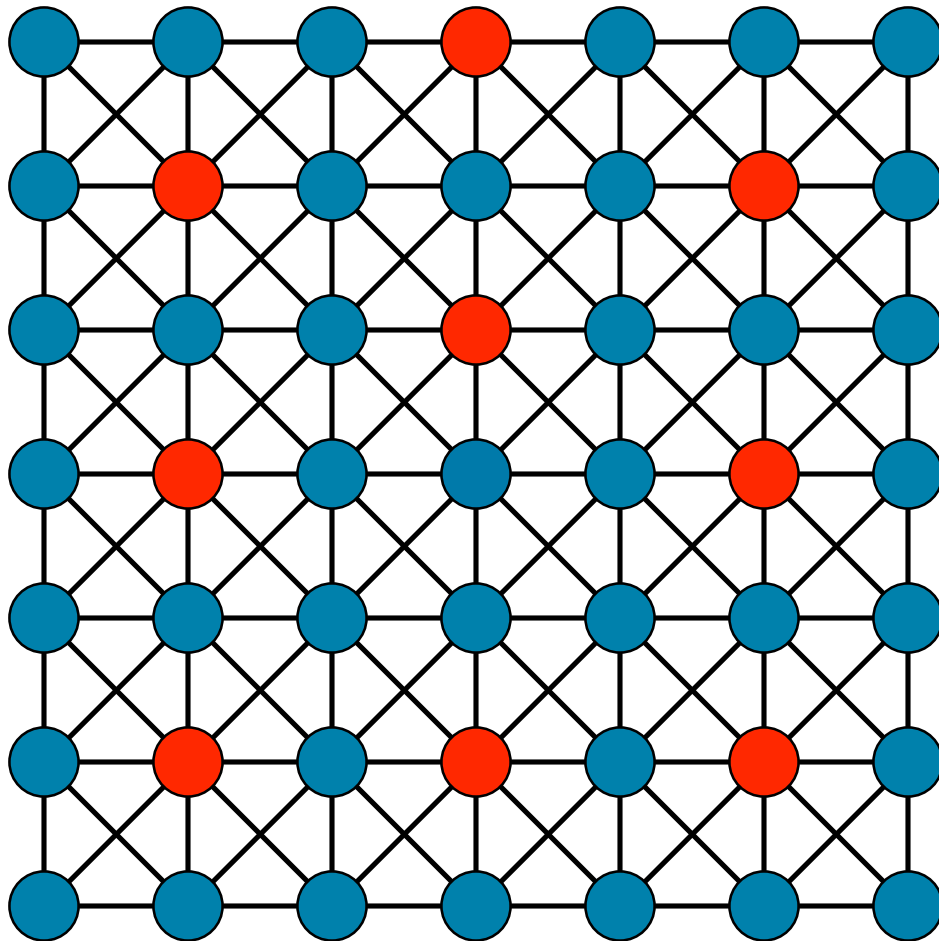➡ remove neighbour edges

PMIS: remove and update 2

→ **select C-pts with maximal measure locally**

→ **make neighbours F-pts**

→ **remove neighbour edges**

PMIS: final grid

→ **select C-pts with maximal measure locally**

→ **make neighbour F-pts**

→ **remove neighbour edges**

→ **parallel algorithm**

PMIS coarsening: reduce complexity

- *finite difference Laplacian (CLJP-PMIS+GMRES)*

	dof	C_{op}	t_{tot}
2D	120^2	4.16	0.22
	120^2	1.90	0.24
3D	100^3	25.94	129.42
	100^3	2.36	27.68
4D	20^4	127.5	88.39
	20^4	2.95	4.31
5D	9^5	256.9	73.92
	8^5	3.14	0.91

Parallel PMIS results: 7-point finite difference Laplacian in 3D, 40^3 dof per proc

CLJP and **PMIS-GMRES(10)**

proc	C_{op}	Iter	t_{total}
1	14.39	6	3.35
512	17.02	10	35.83
1331	17.19	10	46.25
1	*2.32*	*13*	*1.28*
512	*2.37*	*25*	*12.77*
1331	*2.37*	*28*	*17.99*

Convergence problems on PMIS-coarsened grids

- *PMIS coarsening works well for many problems*
- *for some problems, too many iterations are necessary because interpolation is not accurate enough ("not enough C-points")*
- *one solution: add C-points (CLJP…)*
- *other solution: use distance-two C-points for interpolation = long-range interpolation*
  - *Stueben's multipass interpolation*
  - *F-F interpolation*

Convergence problems

- *3D elliptic PDE with jumps in coefficient a*

$$(au_x)_x + (au_y)_y + (au_z)_z = 1$$

- *1000 processors, 40^3 dof/proc*

	t_{tot}	C_{op}	Iter
CLJP	52.48	17.00	17
PMIS	211.79	2.40	686

- *remedy: improve interpolation used with PMIS*

F-F interpolation
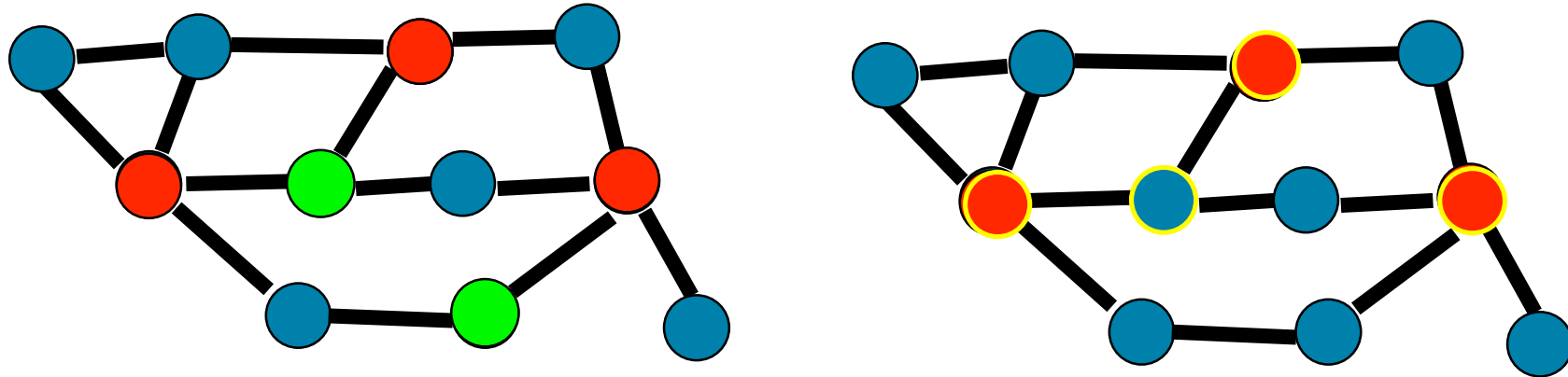
- *when strong F-F connection without a common C-point is detected, do not add C-point, but extend interpolation stencil to distance-two C-points*

- *no C-points added, but larger interpolation stencils*

results using long-range interpolation

- *3D elliptic PDE with jumps in coefficient a*

$$(au_x)_x + (au_y)_y + (au_z)_z = 1$$

- *1 processor, AMG+GMRES, 80^3 dof*

	t_{tot}	C_{op}	Iter
CLJP	48.0	21.54	7
PMIS	94.6	2.46	188
PMIS + mp	13.7	2.47	21
PMIS + F-F	21.4	4.90	9

Conclusions and future work

- *PMIS leads to reduced, scalable complexities for large multi-D problems on parallel computers*

- *for difficult problems, nearest-neighbour interpolation is not sufficient on PMIS grids*

- *long-range interpolation improves convergence*

- *multipass and F-F interpolations were studied*

Top 500 Supercomputer list (November 2004)

Rank	Site	Computer	Country	TeraFlops	Processors
1	Lawrence Livermore National Laboratory	IBM BlueGene/L	US	135	65,536
2	NASA/Ames Research Center/NAS	SGI Altix	US	51	10,160
3	The Earth Simulator Center	NEC Earth-Simulator	Japan	35	5,120
4	Barcelona Supercomputer Center	IBM eServer	Spain	20	3,564
5	Lawrence Livermore National Laboratory	Intel Itanium2	US	19	4,096
6	Los Alamos National Laboratory	ASCI Q - HP AlphaServer	US	13	8,192
7	Virginia Tech	1100 Dual 2.3 GHz Apple XServe	US	12	2,200
8	IBM - Rochester	IBM BlueGene/L	US	11	8,192
9	Naval Oceanographic Office	IBM eServer	US	10	2,944
10	NCSA	Dell P4 Xeon	US	10	2,500
11	ECMWF	IBM eServer	UK	10	2,176
12	ECMWF	IBM eServer	UK	10	2,176
...					
17	Shanghai Supercomputer Center	Dawning 4000A, Opteron	China	8	2,560
18	Los Alamos National Laboratory	LNX Opteron	US	8	2,816
19	Lawrence Livermore National Laboratory	MCR Linux Cluster Xeon	US	8	2,304
20	Lawrence Livermore National Laboratory	ASCI White, IBM SP Power3	US	7	8,192

- *scalable results were presented for MCR (#19), 2,000 procs*

- *next target: Blue Gene/L (#1), 65,000 procs (will be 130,000 soon)*

LLNL Blue Gene/L

- *our code currently runs on LLNL Blue Gene/L*

- *one preliminary result, on 8,000 processors: PMIS works, CLJP runs out of memory...*

- *more tests to follow...*

- *scalability up to 130,000 processors?*