Particle Simulations with OpenACC: Speedup and Scaling

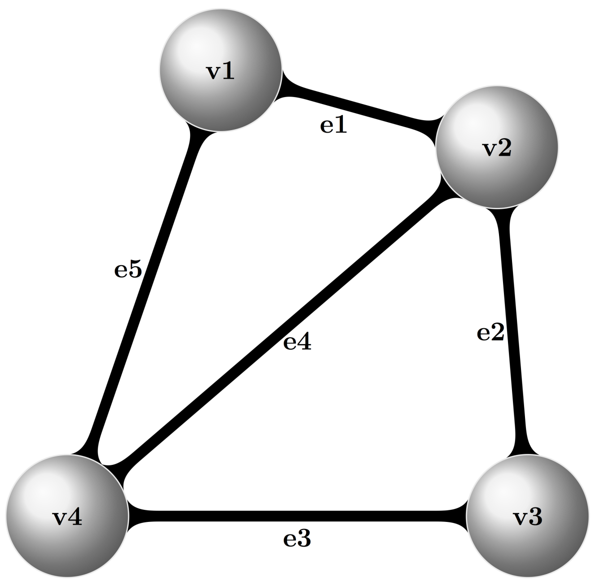
Università della Svizzera italiana | Institute of Computational Science ICS

Samuel A. Cruz Alegria, Alessandra M. de Felice, Hrishikesh R. Gupta

OpenACC

For a graph $\mathcal{G}(V, E)$ with n vertices and m edges:

- Incidence matrix: $\boldsymbol{A} \in \mathbf{R}^{m \times n}$,
- Graph Laplacian matrix: $\boldsymbol{L} \in \mathbf{R}^{n \times n}$,
- Vertex potentials: $x_i = \begin{cases} 1, & i \in V_k, \\ 0, & i \in \overline{V}_k. \end{cases}$



$$\boldsymbol{A} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{L} = \boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

Calculate an edge separator using the Fiedler eigenvector of $\mathbf{L} \in \mathbf{R}^{n \times n}$.

2-Laplacian partitioning

$$\min_{V_k \subset V} \frac{\|\boldsymbol{A}\hat{\mathbf{x}}\|_2^2}{\|\hat{\mathbf{x}}\|_2^2} \approx \min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{\|\boldsymbol{A}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} = \min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{\mathbf{x}^\top \boldsymbol{L}\mathbf{x}}{\mathbf{x}^\top\mathbf{x}} = \lambda_i^{(2)}$$

subject to $\mathbf{1}^\top\mathbf{x} = 0$

Feasible Projection

$$\hat{\mathbf{x}} = \mathbf{x} - \frac{\mathbf{1}^\top\mathbf{x}}{n}$$

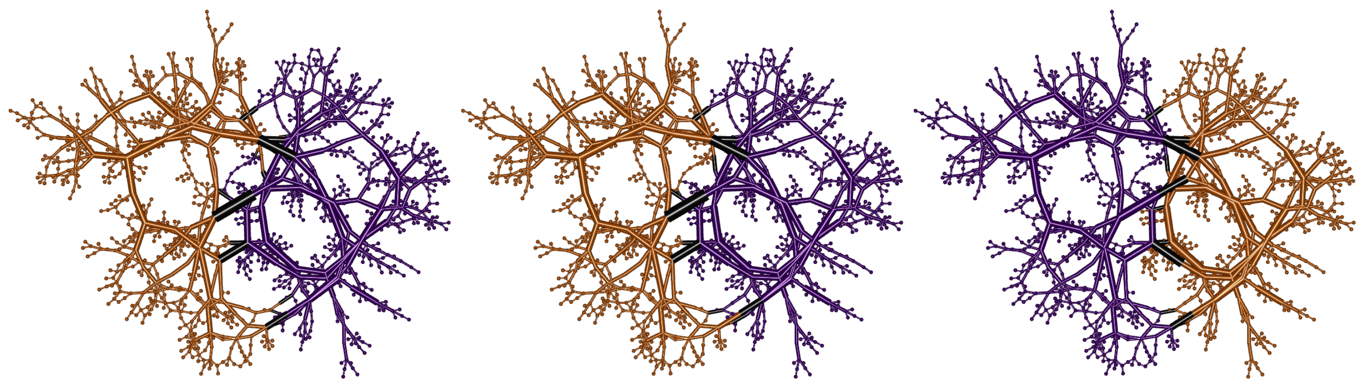
p-Laplacian partitioning

$$\min_{V_k \subset V} \frac{\|\boldsymbol{A}\hat{\mathbf{x}}\|_p^p}{\|\hat{\mathbf{x}}\|_p^p} \approx \min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{\|\boldsymbol{A}\mathbf{x}\|_p^p}{\|\mathbf{x}\|_p^p} = \min_{\mathbf{x} \in \mathbf{R}^n} \frac{(\boldsymbol{A}\mathbf{x})^\top \phi_p(\boldsymbol{A}\mathbf{x})}{\mathbf{x}^\top \phi_p(\mathbf{x})} = \lambda_i^{(p)}$$

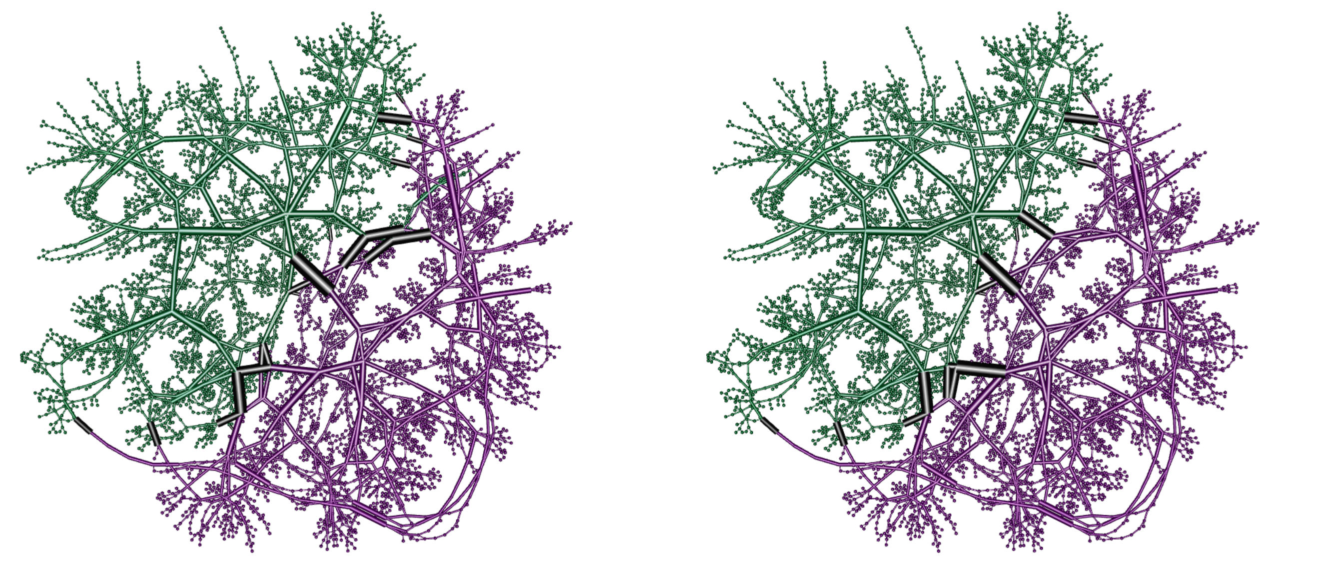
subject to $\mathbf{1}^\top\phi_p(\mathbf{x}) = 0$

$$\phi_p(x_i) = |x_i|^{p-2}x_i, \ i = 1, \ldots, n$$
$$\phi_p^{-1}(x_i) = |x_i|^{\frac{1}{p-1}}\operatorname{sign}(x_i)$$
$$\hat{\mathbf{x}}_p = \phi_p^{-1}\left(\phi_p(\mathbf{x}) - \frac{\mathbf{1}^\top\phi_p(\mathbf{x})}{n}\right)$$

OpenACC Examples and Uses



Case	Nodes	Edges	**METIS**	$b_r^2(\%)$	**p-Lap(METIS%)**	$b_r^2(\%)$	KaHIP	$b_r^2(\%)$	**p-Lap(KaHIP%)**	$b_r^2(\%)$
1354pegase	1,354	1,991	**20**	0.0	**17 (15.0)**	1.3	18	1.5	**16 (11.1)**	3.0



Case	Nodes	Edges	METIS	$b_r^2(\%)$	p-Lap(METIS%)	$b_r^2(\%)$	**KaHIP**	$b_r^2(\%)$	**p-Lap(KaHIP%)**	$b_r^2(\%)$
6495rte	6,495	9,019	35	0.0	34 (2.9)	0.2	**32**	5.4	**25 (21.9)**	5.9

Particle Simulations

Algorithm 1 p-Laplacian Bisection

Input: $\mathbf{x}_0$ ▷ METIS or KaHIP bisection
Output: $\mathbf{x}_p^{\min}$ ▷ p-Laplacian bisection

function pLaplacian($\boldsymbol{A}, \mathbf{x}_0, b^{\max}, \beta$, max_it)
  $r_c^{\min} \leftarrow \mathrm{RCCut}(\mathbf{x}_0)$
  $p = 2, \quad \mathbf{x} = \mathbf{x}_0$
  for k=0:max_it **do**
    $p_k = 1 + e^{-\beta k/\text{max\_iters}}$
    $\mathbf{x}_k^{\min} \leftarrow \mathrm{pLaplacianDescent}(\boldsymbol{A}, p_k)$
  end for
  return $\mathbf{x}_p^{\min}$
end function

Algorithm 2 p-Laplacian Descent

Input: $\mathbf{x}_0$ ▷ approximation of the p-eigenvector
Output: $\mathbf{x}_p^{\min}$ ▷ p-Laplacian bisection

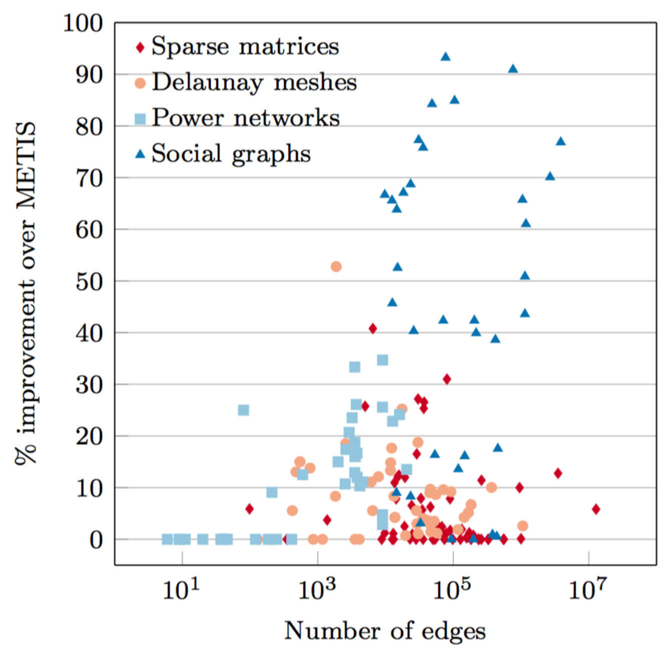
function pLaplacianDescent($\boldsymbol{A}, \mathbf{x}_0, p$)
  while not converged **do**
    $\mathbf{x} \leftarrow \hat{\mathbf{x}}_p$
    $r_c = \mathrm{RCCut}(\mathbf{x})$
    $b_r = \mathrm{ImBal}(\mathbf{x})$
    if $r_c < r_c^{\min}$ **and** $b_r < b_r^{\max}$ **then**
      $\mathbf{x}_p^{\min} \leftarrow \mathbf{x}$ ▷ save best solution
      $r_c^{\min} \leftarrow r_c$ ▷ and minimum cut
    end if
    $\mathbf{g} \leftarrow \nabla f(\mathbf{x})$
    $\alpha \leftarrow \operatorname{argmin}_{\alpha} f\left(\phi_p^{-1}(\mathbf{x} - \alpha\mathbf{g})\right)$
    $\mathbf{x} \leftarrow \phi_p^{-1}(\mathbf{x} - \alpha\mathbf{g})$
  end while
  return $\mathbf{x}_p^{\min}$
end function

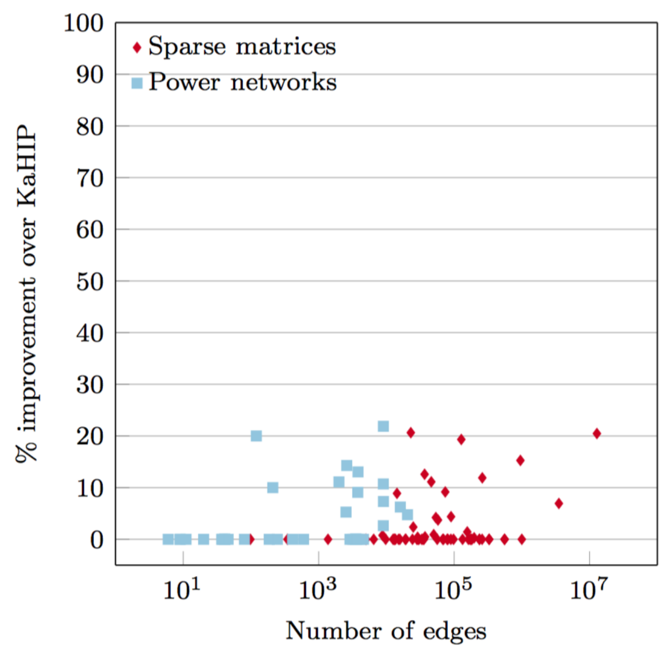
- Ratio Cheeger cut: $\mathrm{RCCut}(V_k, \overline{V}_k) = \frac{\mathrm{cut}(V_k, \overline{V}_k)}{\min\{|V_k|, |\overline{V}_k|\}}$, with $\mathrm{cut}(V_k, \overline{V}_k) = \|\boldsymbol{A}\mathbf{x}\|_p^p$.
- Node imbalance: $b_r^2 = \frac{||V_2| - |\overline{V}_2||}{|V|}$.
- Reduced computational complexity: $O(n^3) \to O(m)$.

Particle Simulations integrated with OpenACC

- Consistent and valid improvement over a METIS or KaHIP partition.
- The level of improvement achieved depends on the structure of the graph in question and the optimality or near optimality of the original cut

From a total of 186 graphs initialized with a METIS cut, improvements over 10% were observed for 47.3% of them, while for the 102 graphs initialized with a KaHIP partition improvements over 10% were observed for 15.7% of them.





Parallel particle simulations

Motivation - ICS and Master Programme

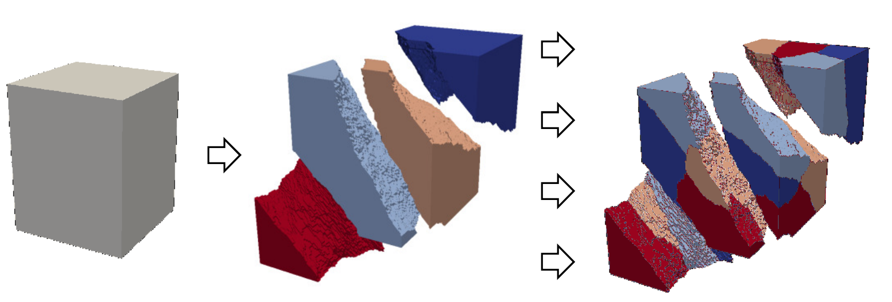
The bisection of a finite-element mesh on an XC50 compute node of Piz Daint at the Swiss National Supercomputing Centre indicates that the p-Laplacian algorithm is suitable for parallel computation and is able to utilize the capabilities of modern computer hardware.

15,360,000 elements, 2,618,021 nodes

CPU	Peak FLOPS	(Peak MEM.)	Elapsed time (s)
Intel Xeon	499.2 GFLOPS	68 GB/s	1862.3 (1 core)
E5-2690 v3 CPU	—	—	284.8 (12 cores)
Nvidia P100 GPU	4.7 TFLOPS	732 GB/s	28.2

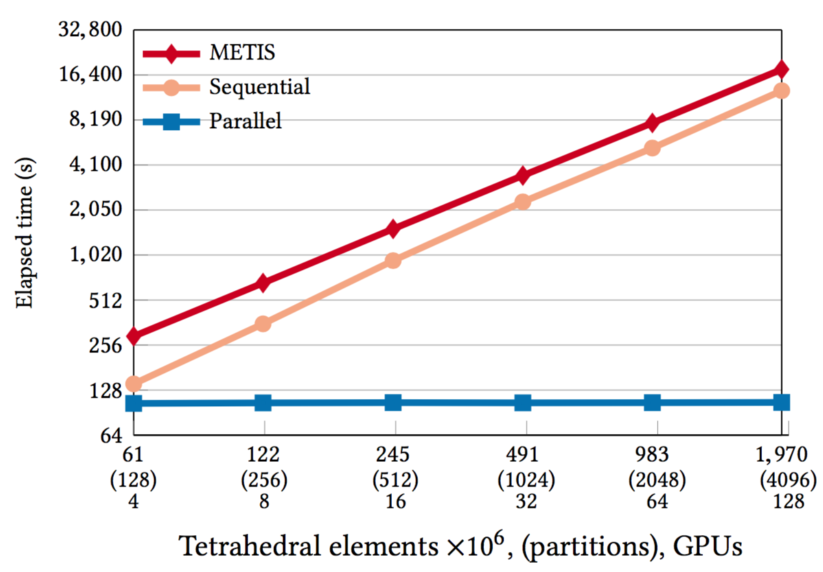
ICS Cluster

Method	edgecut	(METIS%)	$\max_{i \leq 128} \lvert V_i^{128} \rvert$	$b_r^{128}(\%)$
METIS	600,450	—	120,000	0
p-Laplacian	558,492	7.5	120,939	0.78
Hybrid	570,240	5.3	120,660	0.55



Visualisation and Performance

The elapsed time is measured on SGI UV300 for METIS and part 1 of the hybrid method, and P100 GPUs on Piz Daint XC50 nodes. The largest finite-element mesh partitioned, consisting of 1.9 billion tetrahedra, corresponds to an application using 77% of the Piz Daint system (4,096 compute nodes).



T. Simpson, D. Pasadakis, D. Kourounis, K. Fujita, T. Yamaguchi, T. Ichimura, and O. Schenk, "Balanced graph partition refinement using the graph p-Laplacian," in *Proceedings of the ACM Platform for*