A High-Throughput Solver for Marginalized Graph Kernels on GPU

Yu-Hang Tang, Oguz Selvitopi, Doru Popovici, Aydin Buluc
Computational Research Division
Lawrence Berkeley National Laboratory

IPDPS 2020
May 21, 2020

Content

PART 1

Marginalized graph kernel for learning on graphs

PART 2

GPU-accelerated high throughput solver

SUMMARY

PART 1

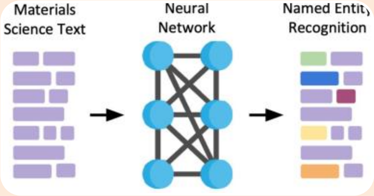
Marginalized graph kernel for learning on graphs

Scientific machine learning is key to DOE technological advances

Scientific machine learning (SciML) is a core component of artificial intelligence (AI) *and a computational technology that can be trained, with scientific data, to augment or automate human skills.* ***Across the Department of Energy (DOE)****, scientific machine learning (SciML) has the potential to* ***transform science and energy research****.*

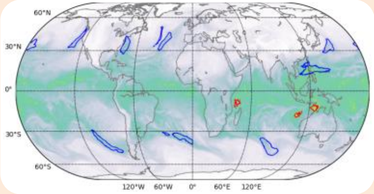
DOE Basic Research Needs Workshop for Scientific Machine Learning:
Core Technologies for Artificial Intelligence 2019

The successes of scientific machine learning have concentrated on select forms of data



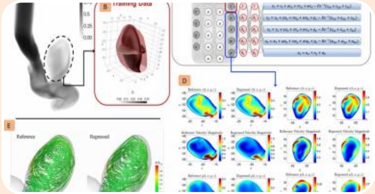
Literature information extraction

- Text data
- Linear sequence
- Weston et al. 2019



Climate analytics

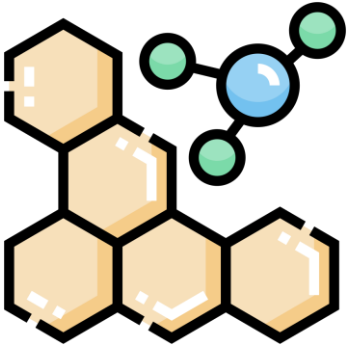
- Grid data
- Real values
- Kurth et al. 2018



Fluid Mechanics

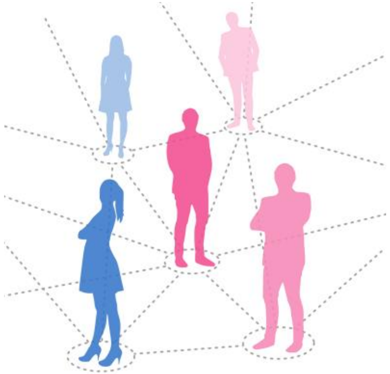
- Mesh data
- Real values
- Raissi et al., 2020

Many scientific data and representations are beyond mere images or linear sequences

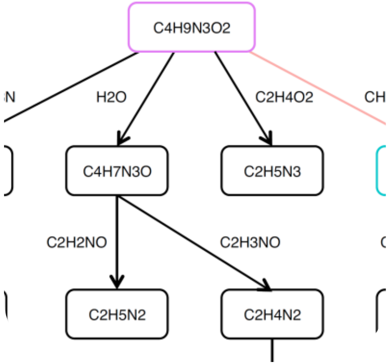


Molecules

Road network



Social network



Fragmentation tree

Variable in size

Non-sequential

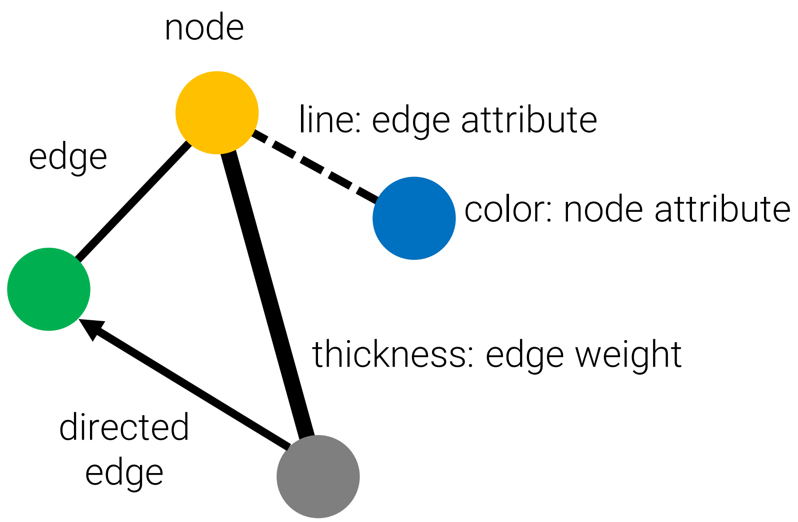
Mixed continuous/discrete DOFs

Existing solutions often resort to pixelating the data.

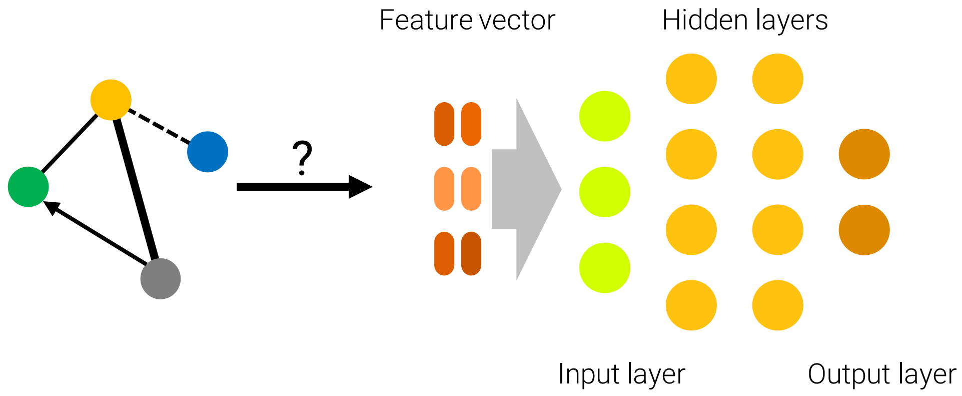
Some icons made by Freepik from www.flaticon.com

Graph is a powerful format for scientific data, but machine learning on graphs takes extra effort

- A graph is a structure that contains objects of pairwise relationships



- Most existing ML methods work on feature vectors, images, and sequences only.

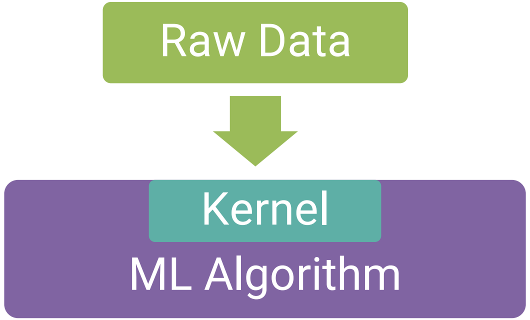


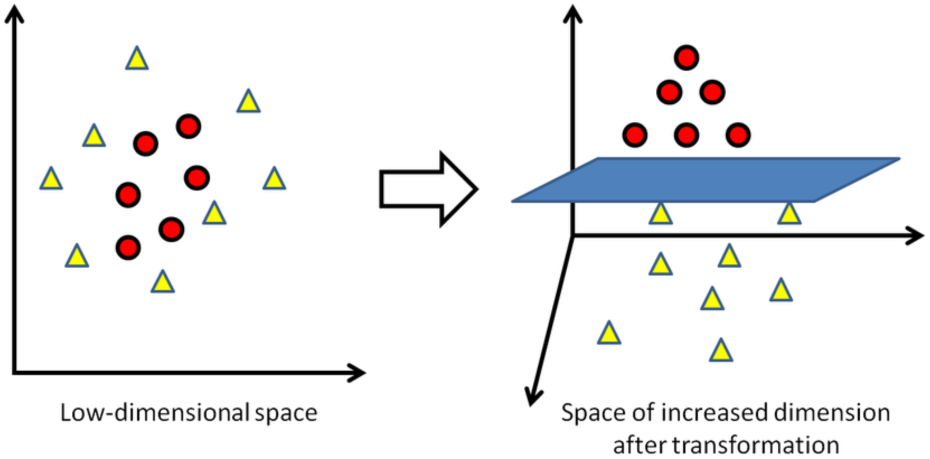
Kernel method in machine learning: what, why, and how

A kernel is a function that
- Implicitly transforms raw data into high-dimensional feature vectors via a **feature map**; and then
- Returns an **inner product** between the feature vectors.
- Must be **positive-definite.**

A kernel is useful for
- **Factor out** knowledge on data representation from downstream algorithms,
- Exploit **infinite dimensionality and nonlinear** feature spaces.

Kernels are used in
- Support vector machine (SVM), Gaussian process regression (GPR), Kernel principal component analysis (kPCA), etc.

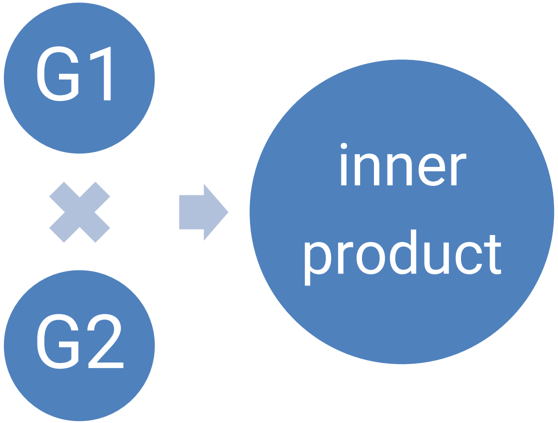




Many ML algorithms have a kernelized counterpart

$w^T\phi(x_n) + b$	**SVM**	$\sum a_n t_n k(x, x_n) + b$
$w^T\phi(x_n)$	**Ridge Regression**	$k(x)^T(K + \lambda I)^{-1}t$
$\mathrm{eig}\left(\Phi^{\mathrm{T}}\Phi\right)$	**PCA**	$\mathrm{eig}\left[k(x_i, x_j)\right]$
Euclidean distance	**Clustering**	Kernel-induced distance
Project on random vectors	**Random Projection**	Project on random samples

Graph kernels are kernels that act on graphs



Graph kernels	Histogram	Limited-size subgraphs [Ahmed et al. 2015]
		Statistical moments [Debnath et al. 1991]
	Random walk	Exponential [Vishwanathan, 2010]
		Geometric [Vishwanathan, 2010]
		Marginalized [Kashima et al., 2003]
	Weisfeiler-Lehman	[Shervashidze et al. 2011]
		[Morris et al. 2017]
	Misc.	Shortest-path [Borgwardt and Kriegel, 2005]
		Spanning tree [Ramon and Gärtner, 2003]

The marginalized graph kernel can seamlessly handle diverse types of graphs

- Definition: the inner product between two graphs is the statistical average of the inner product of simultaneous random walk paths on the two graphs.

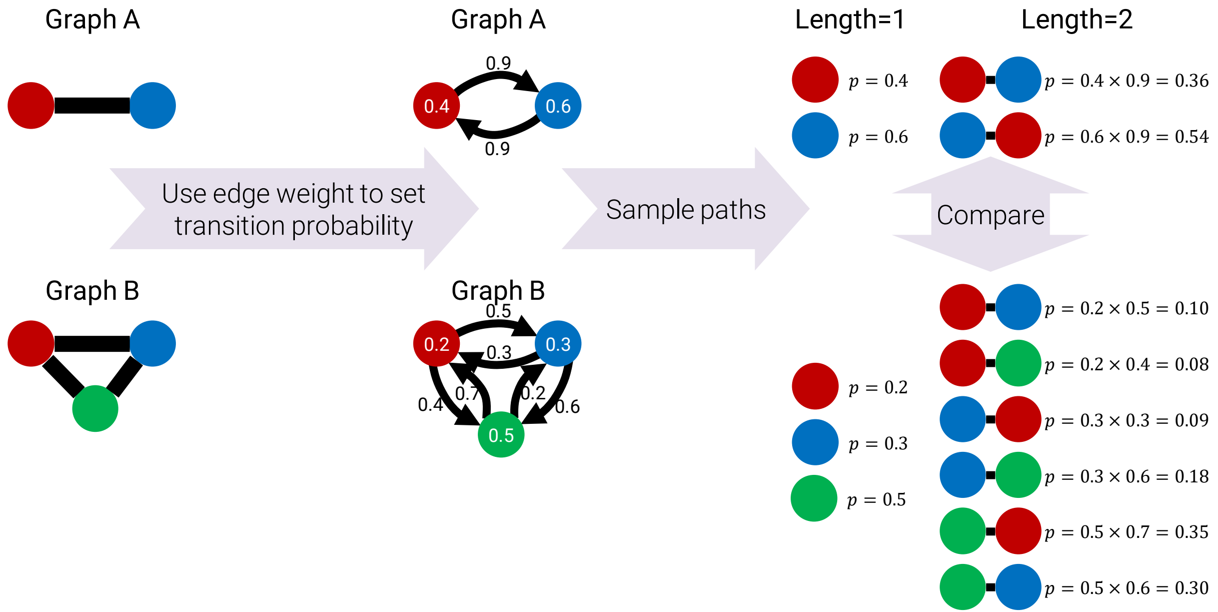
Step 1
Define random walks

$$P = D^{-1} \cdot A$$

P: transition matrix

D: degree matrix

A: adjacency matrix



The marginalized graph kernel can seamlessly handle diverse types of graphs

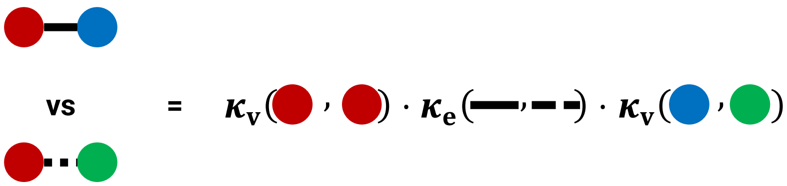
- Definition: the inner product between two graphs is the statistical average of the inner product of simultaneous random walk paths on the two graphs.

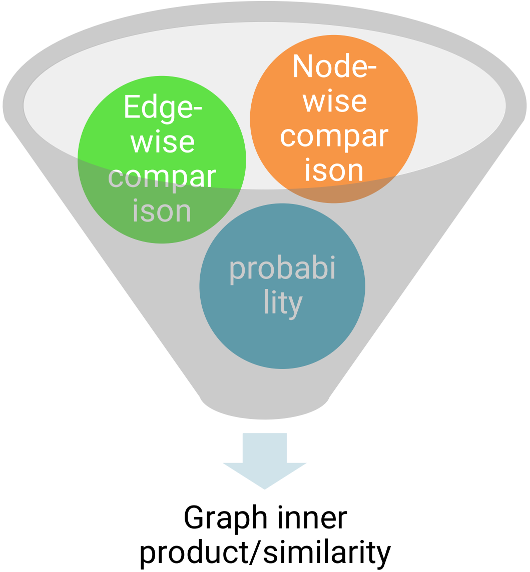
Step 2
Averaging path similarities

Path similarity defined as product of base kernel evaluations

κ_v: base kernel for nodes

κ_e: base kernel for edges





Wider adoption of the marginalized graph kernel was hindered due to practical challenges

$$K(G,G') = \sum_{l=1}^{\infty}\sum_{\boldsymbol{h}}\sum_{\boldsymbol{h}'} p_{\mathrm{s}}(h_1)p'_{\mathrm{s}}(h'_1)\boldsymbol{K_v}\left(v_{h_1}, v'_{h'_1}\right)\prod_{i=2}^{l} p_{\mathrm{t}}(h_i|h_{i-1})p_{\mathrm{q}}(h_l)\prod_{j=2}^{l} p'_{\mathrm{t}}(h'_j|h'_{j-1})p'_{\mathrm{q}}(h'_l)\prod_{k=2}^{l}\boldsymbol{K_e}\left(e_{h_{k-1}h_k}, e_{h'_{k-1}h'_k}\right)\boldsymbol{K_v}\left(v_{h_k}, v'_{h'_k}\right)$$

Cost of computation could be high

- direct summation is **intractable**

Efficient training involving composite base kernels K_{v}, K_{e} is non-trivial

- **Analytic derivative** of the kernel is difficult to derive and implement

Linear algebra reformulation simplifies computations and reveals opportunities for optimization

$$K(G,G') = \sum_{l=1}^{\infty}\sum_{\boldsymbol{h}}\sum_{\boldsymbol{h}'} p_s(h_1)p_s'(h_1')\boldsymbol{K_v}\left(v_{h_1}, v'_{h'_1}\right)\prod_{i=2}^{l} p_t(h_i|h_{i-1})p_q(h_l)\prod_{j=2}^{l} p_t'(h_j'|h_{j-1}')p_q'(h_l')\prod_{k=2}^{l}\boldsymbol{K_e}\left(e_{h_{k-1}h_k}, e_{h'_{k-1}h'_k}\right)\boldsymbol{K_v}\left(v_{h_k}, v'_{h'_k}\right)$$

- According to Kashima & Tsuda, the above computation can be simplified into

$$K(G,G') = \sum_{h}\sum_{h'} s(h_1,h_1')R_\infty(h_1,h_1')$$

where

$$s(h_1,h_1') = p_s(h_1)p_s'(h_1')$$

$$R_\infty(h_1,h_1') = r_1(h_1,h_1') + \sum_{i,j} t(i,j,h_1,h_1')R_\infty(h_1,h_1')$$

with

$$t(i,j,h_1,h_1') = p_t(i|h_1)p_t'(j|h_1')K_v\left(v_i,v_j'\right)K_e\left(e_{ih_1}, e_{jh_1'}\right)$$

- We showed that the formulation is equivalent to the following tensor product linear system:

$$K(G,G') = \boldsymbol{p}_\times \cdot \mathbf{R}_\infty$$

where R_∞ can be solved from

$$[\mathbf{D}_\times \mathbf{V}_\times^{-1} - \mathbf{A}_\times \odot \mathbf{E}_\times]\,\mathbf{R}_\infty = \mathbf{D}_\times\,\mathbf{q}_\times.$$

$$p_{\times ij} = p_s(i)p_s'(j), q_{\times ij} = p_q(i)p_q'(j)$$

$$\mathrm{diag}(D_\times)_{ij} = \deg(v_i)\deg\left(v_j'\right)$$

$$\mathrm{diag}(V_\times)_{ij} = K_v\left(v_i,v_j'\right)$$

$$A_{\times ijkl} = w_{ij}w_{kl}'$$

$$E_{\times ijkl} = K_e(e_{ij},e_{kl}')$$

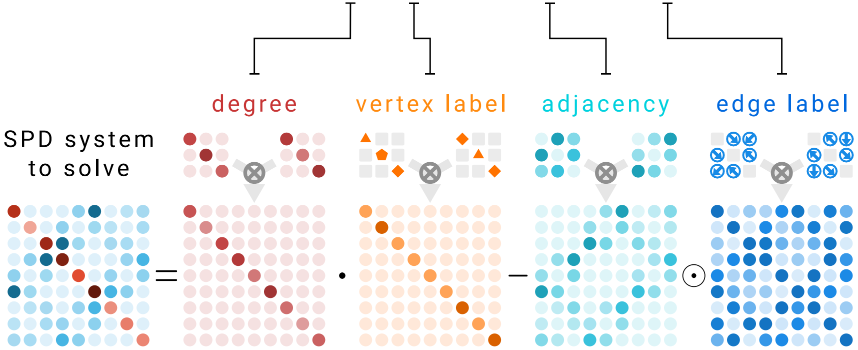
multi-index: ij: element at $i\cdot n' + j$ $ijkl$: element at (ij,kl)

Linear algebra reformulation simplifies computations and reveals opportunities for optimization

$$K(G,G') = \sum_{l=1}^{\infty} \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h}'} p_{\mathrm{s}}(h_1) p'_{\mathrm{s}}(h'_1) K_{\mathrm{v}}\left(v_{h_1}, v'_{h'_1}\right) \prod_{i=2}^{l} p_{\mathrm{t}}(h_i|h_{i-1}) p_{\mathrm{q}}(h_l) \prod_{j=2}^{l} p'_{\mathrm{t}}(h'_j|h'_{j-1}) p'_{\mathrm{q}}(h'_l) \prod_{k=2}^{l} K_{\mathrm{e}}\left(e_{h_{k-1}h_k}, e_{h'_{k-1}h'_k}\right) K_{\mathrm{v}}\left(v_{h_k}, v'_{h'_k}\right)$$

- The marginalized graph kernel in linear algebra form represents a modified graph Laplacian

$$K(G,G') = \mathbf{p}_{\times}^{\mathsf{T}} \left(\mathbf{D}_{\times} \mathbf{V}_{\times}^{-1} - \mathbf{A}_{\times} \odot \mathbf{E}_{\times}\right)^{-1} \mathbf{D}_{\times} \mathbf{q}_{\times}$$



Tang & de Jong, J Chem Phys, 2019: Prediction of atomization energy using graph kernel and active learning
https://doi.org/10.1063/1.5078640

Linear algebra reformulation simplifies derivation of analytic derivatives

- The gradient of the marginalized graph kernel is crucial for efficient training
- It can be derived using matrix calculus:

$$K(G, G') = \mathbf{p}_\times^\mathrm{T} \left[\mathbf{D}_\times \mathbf{V}_\times^{-1} - \mathbf{A}_\times \odot \mathbf{E}_\times\right]^{-1} \mathbf{D}_\times \, \mathbf{q}_\times$$

Denote

$$\mathbf{Y} = \mathbf{D}_\times \mathbf{V}_\times^{-1} - \mathbf{A}_\times \odot \mathbf{E}_\times$$

Then

$$\frac{\partial K}{\partial \theta} = \mathrm{tr}\left[\frac{\partial K}{\partial \mathbf{Y}} \cdot \frac{\partial \mathbf{Y}}{\partial \theta}\right] = \left(\mathbf{Y}^{-1}\mathbf{p}_\times\right)^T \frac{\partial \mathbf{Y}}{\partial \theta} \left(\mathbf{Y}^{-1}\mathbf{D}_\times \mathbf{q}_\times\right)$$

Differentiation w.r.t. other hyperparameters can be derived similarly.

Marginalized graph kernel has found successful applications in a variety of ML tasks

- Prediction of molecular atomization energy
  - nodes = atoms, edges = interatomic interactions
  - Jump probabilities proportional to edge weights, which decay with interatomic distance

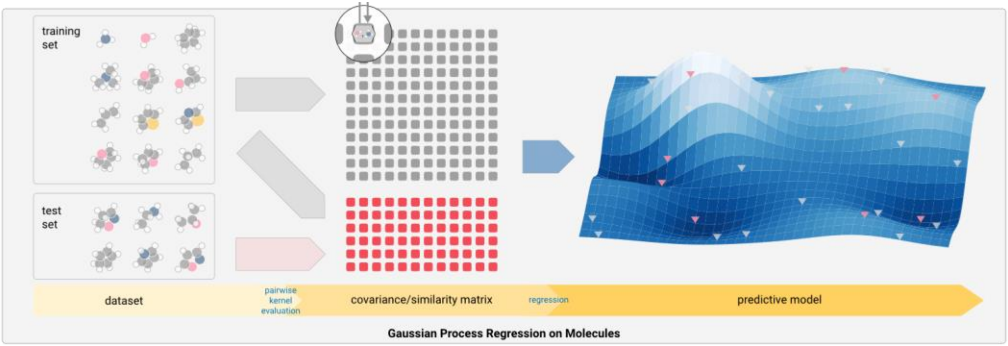
    e.g. $w_{ij} = \left(1 - \frac{r_{ij}}{r_c}\right)^{\mathrm{n}}$

  - Kronecker delta kernel on nodes labeled with chemical elements

    e.g. $\kappa_{\mathrm{v}}(v_1, v_2) = \begin{cases} 1, \text{if } v_1 = v_2 \\ h, \text{otherwise} \end{cases}$ etc.

  - Gaussian kernel on edges labeled by interatomic distance

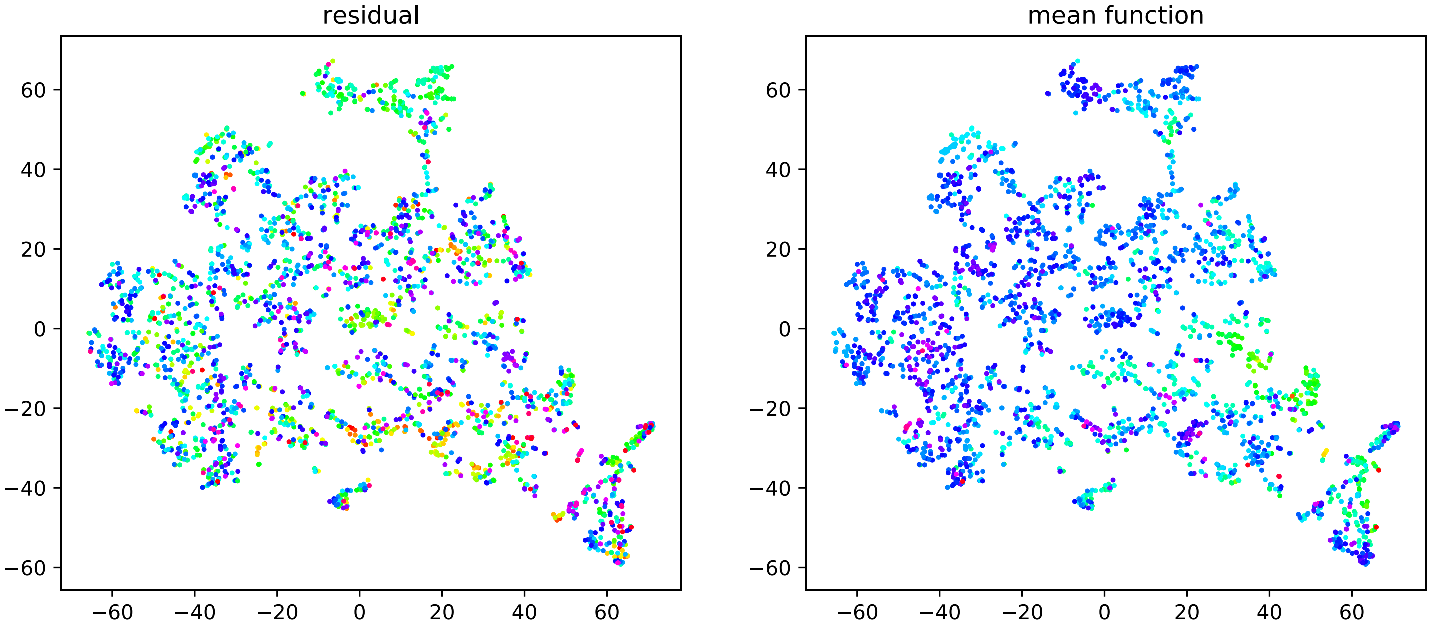
    e.g. $\kappa_{\mathrm{e}}(l_1, l_2) = \exp\left[-\frac{1}{2}\frac{(l_1 - l_2)^2}{\sigma^2}\right]$



Tang & de Jong, J Chem Phys, 2019: Prediction of atomization energy using graph kernel and active learning https://doi.org/10.1063/1.5078640

Marginalized graph kernel has found successful applications in a variety of ML tasks

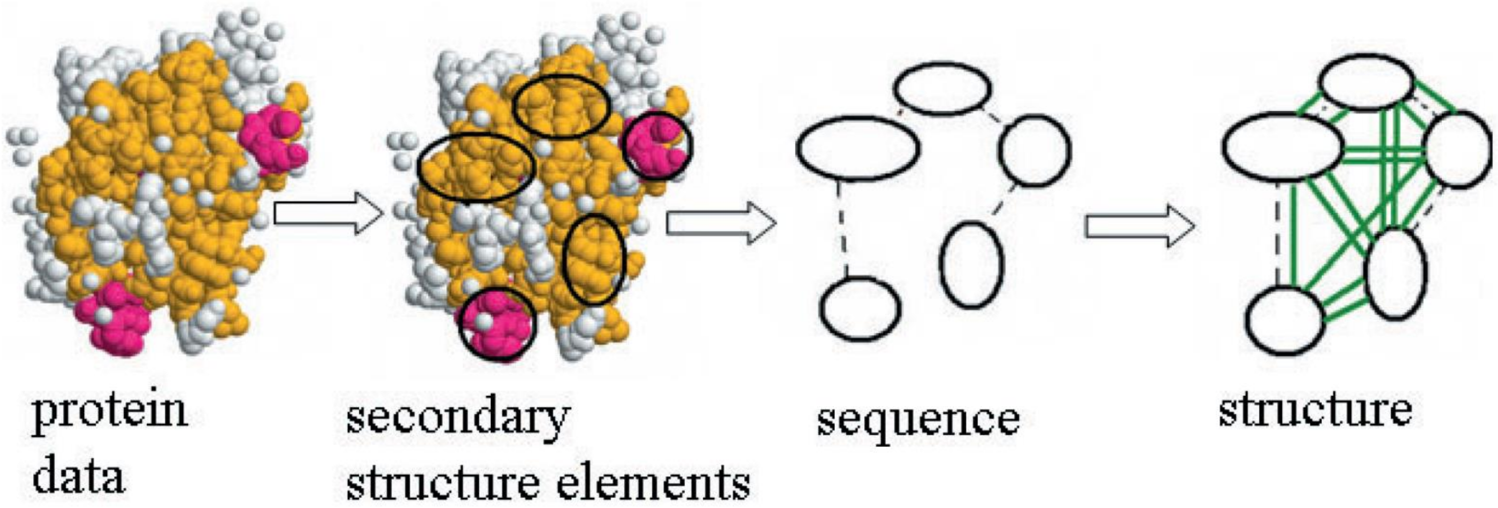
- Quality assurance on noisy chromatography data



Tang et al. Uncertainty Quantification and Outlier Detection on Noisy Data. Manuscript in preparation.

Marginalized graph kernel has found successful applications in a variety of ML tasks

- Protein function prediction



Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., & Kriegel, H.-P. (2005). Protein function prediction via graph kernels. Bioinformatics, 21(suppl_1), i47–i56. https://doi.org/10.1093/bioinformatics/bti1007

Content

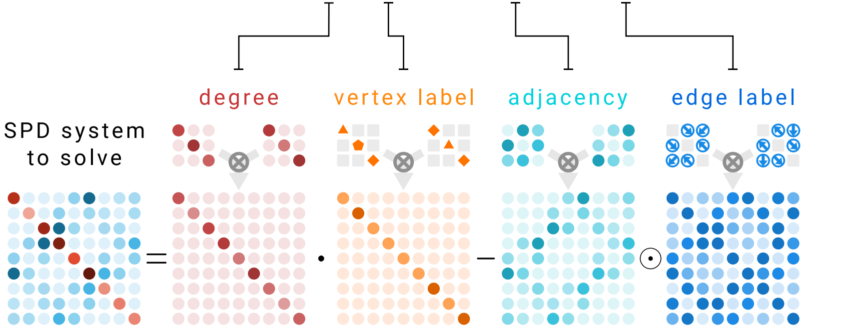
PART 2

GPU-accelerated high throughput solver for marginalized graph kernel

The marginalized graph kernel equation can be efficiently solved using conjugate gradient

- The conjugate gradient algorithm can be used to **iteratively** solve the marginalized graph kernel equation
  - V and E are not necessarily real matrices
  - κ can be complex functions

$$K(G, G') = \mathbf{p}_\times^\mathsf{T} \left(\mathbf{D}_\times \mathbf{V}_\times^{-1} - \mathbf{A}_\times \odot \mathbf{E}_\times\right)^{-1} \mathbf{D}_\times \mathbf{q}_\times$$



function CG4GK($\mathbf{d}, \mathbf{d}', \mathbf{v}, \mathbf{v}', \mathbf{A}, \mathbf{A}', \mathbf{E}, \mathbf{E}'$, $\mathbf{q}, \mathbf{q}'$)

    $\mathbf{M} \leftarrow \mathbf{diag}\left[(\mathbf{d} \otimes \mathbf{d}') \odot (\mathbf{v} \overset{\kappa}{\otimes} \mathbf{v}')^{-1}\right]$

    $\mathbf{x} \leftarrow \mathbf{0}$

    $\mathbf{r} \leftarrow (\mathbf{d} \otimes \mathbf{d}') \cdot (\mathbf{q} \otimes \mathbf{q}')$

    $\mathbf{z} \leftarrow \mathbf{v} \overset{\kappa}{\otimes} \mathbf{v}'$

    $\mathbf{p} \leftarrow \mathbf{z}$

    $\rho \leftarrow \mathbf{r}^\mathsf{T}\mathbf{z}$

    repeat

        $\mathbf{a} \leftarrow (\mathbf{d} \otimes \mathbf{d}') \odot (\mathbf{v} \overset{\kappa}{\otimes} \mathbf{v}')^{-1} \cdot \mathbf{p}$

            $+ (\mathbf{A} \otimes \mathbf{A}') \odot (\mathbf{E} \overset{\kappa}{\otimes} \mathbf{E}') \cdot \mathbf{p}$

        $\alpha \leftarrow \rho / (\mathbf{p}^\mathsf{T}\mathbf{a})$

        $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$

        $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{a}$

        $\mathbf{z} \leftarrow \mathbf{M}^{-1}\mathbf{r}$

        $\rho' \leftarrow \mathbf{r}^\mathsf{T}\mathbf{z}$

        $\beta \leftarrow \rho'/\rho$

        $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$

        $\rho \leftarrow \rho'$

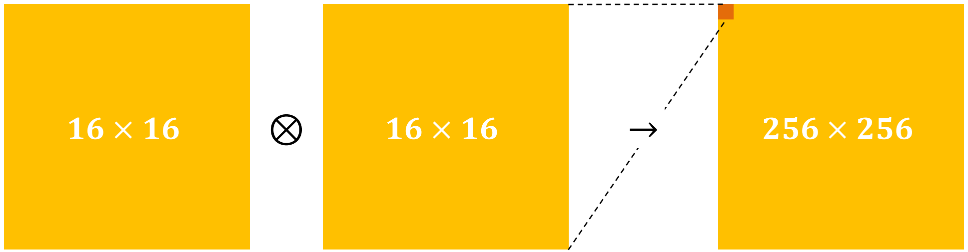
    until $\mathbf{r}^\mathsf{T}\mathbf{r} < \epsilon$

    return $\mathbf{x}$

Naïve CG on precomputed matrices can only handle small graphs

- Due to the tensor product structure of the linear system, memory usage grows in quartic order

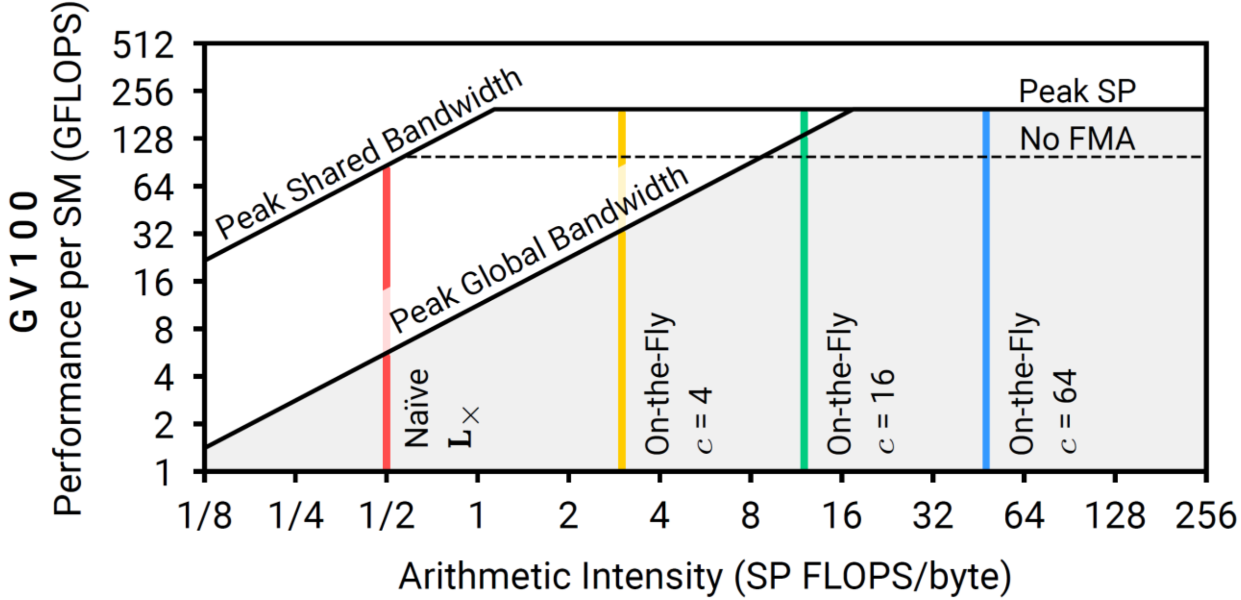
To compute similarity between a pair of 1000-node graphs, a system of 1000000×1000000 (4TB) is involved.



function CG4GK($\mathbf{d}, \mathbf{d}', \mathbf{v}, \mathbf{v}', \mathbf{A}, \mathbf{A}', \mathbf{E}, \mathbf{E}', \mathbf{q}, \mathbf{q}'$)
  $\mathbf{M} \leftarrow \mathbf{diag}\left[(\mathbf{d} \otimes \mathbf{d}') \odot (\mathbf{v} \overset{\kappa}{\otimes} \mathbf{v}')^{-1}\right]$
  $\mathbf{x} \leftarrow \mathbf{0}$
  $\mathbf{r} \leftarrow (\mathbf{d} \otimes \mathbf{d}') \cdot (\mathbf{q} \otimes \mathbf{q}')$
  $\mathbf{z} \leftarrow \mathbf{v} \overset{\kappa}{\otimes} \mathbf{v}'$
  $\mathbf{p} \leftarrow \mathbf{z}$
  $\rho \leftarrow \mathbf{r}^\mathsf{T}\mathbf{z}$
  repeat
    $\mathbf{a} \leftarrow (\mathbf{d} \otimes \mathbf{d}') \odot (\mathbf{v} \overset{\kappa}{\otimes} \mathbf{v}')^{-1} \cdot \mathbf{p}$
      $+(\mathbf{A} \otimes \mathbf{A}') \odot (\mathbf{E} \overset{\kappa}{\otimes} \mathbf{E}') \cdot \mathbf{p}$
    $\alpha \leftarrow \rho / (\mathbf{p}^\mathsf{T}\mathbf{a})$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{a}$
    $\mathbf{z} \leftarrow \mathbf{M}^{-1}\mathbf{r}$
    $\rho' \leftarrow \mathbf{r}^\mathsf{T}\mathbf{z}$
    $\beta \leftarrow \rho'/\rho$
    $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
    $\rho \leftarrow \rho'$
  until $\mathbf{r}^\mathsf{T}\mathbf{r} < \epsilon$
  return $\mathbf{x}$

Naïve CG on precomputed matrices is also memory-bound on GPUs

- NVIDIA Volta GPU requires more than 16 FLOPS per byte (64 FLOPS per float) arithmetic intensity to achieve peak performance



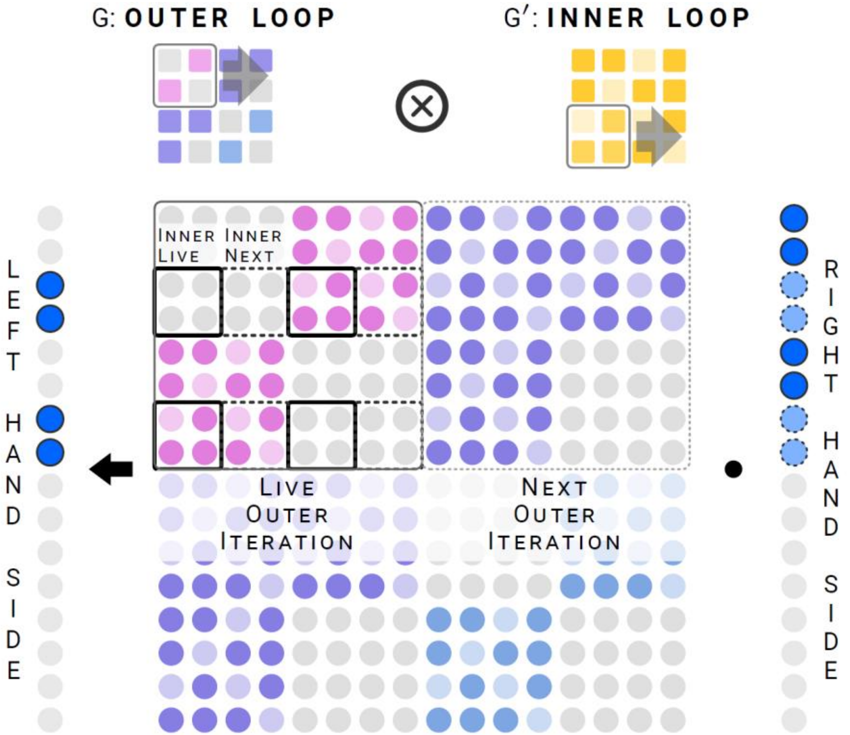
On-the-fly Kronecker matrix-vector multiplication (XMV) can overcome storage and memory bandwidth difficulties

On-the-fly Kronecker matrix-vector multiplication (OTF XMV)

- Regenerates the product linear system on the fly by streaming 8-by-8 submatrices (tiles).
  - Tiles staged in shared memory.
- Trade FLOPS for GB/s, but asymptotic arithmetic complexity stays the same.

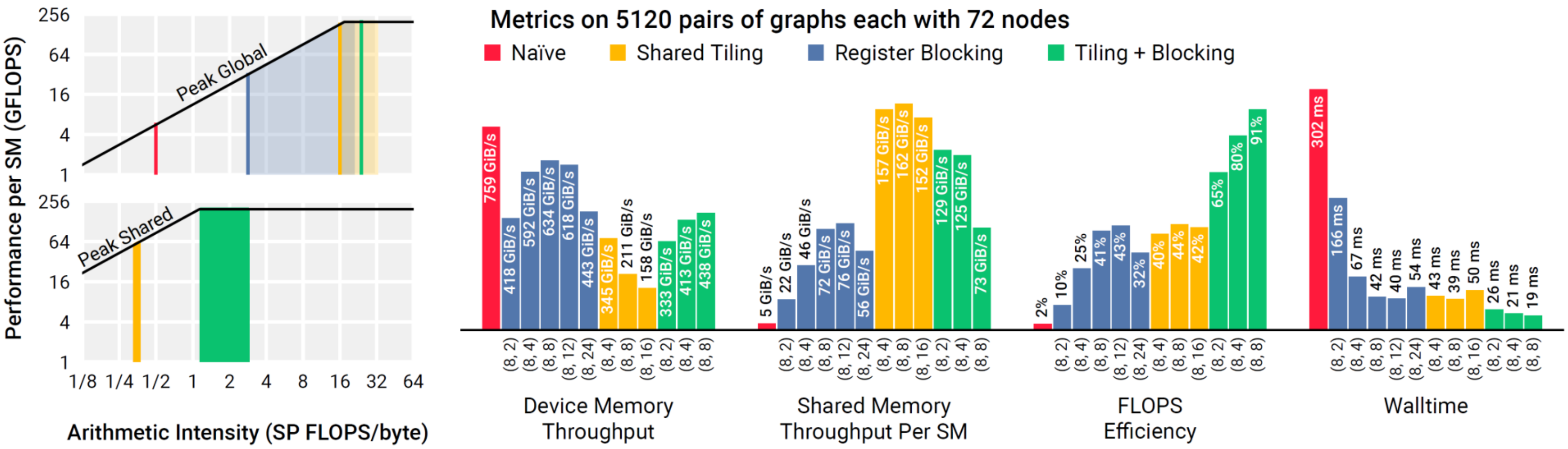
```
repeat
    a ← (d ⊗ d′) ⊙ (v ⊗^κ v′)^{-1} · p
        +(A ⊗ A′) ⊙ (E ⊗^κ E′) · p
    α ← ρ/(p^T a)
    x ← x + αp
```

Tang, Selvitopi, Popovici & Buluc, IPDPS 2020: A High-Throughput Solver for Marginalized Graph Kernels on GPU https://arxiv.org/abs/1910.06310



OTF XMV achieves much higher FLOPS on dense graphs

- Microbenchmark on V100 with a dot product base kernel

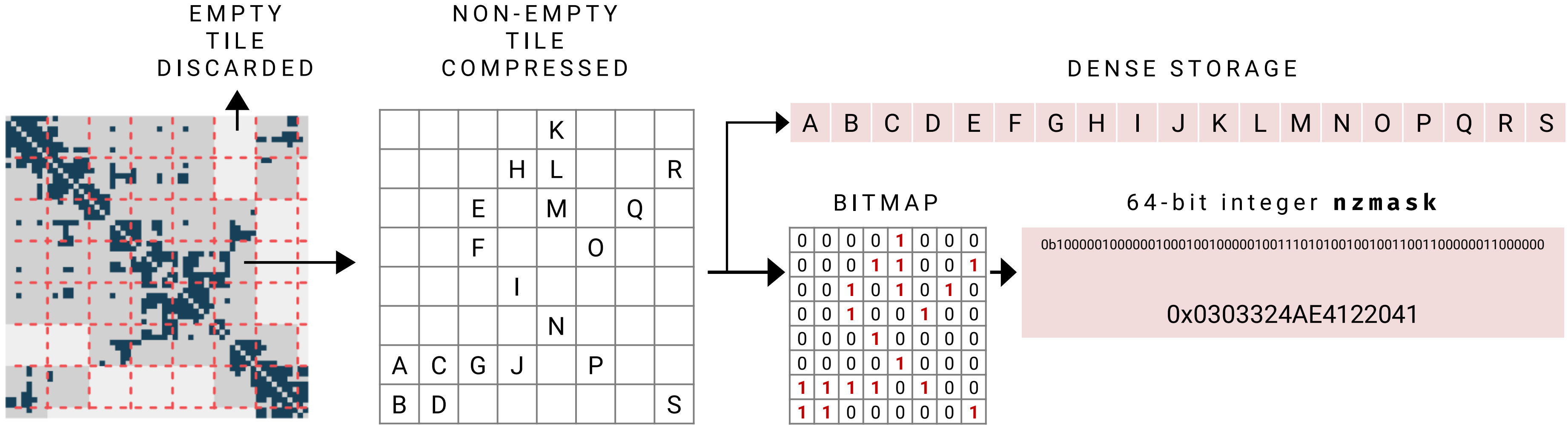


A 2-level hierarchical sparse matrix format ensures efficient memory usage

2-Level sparsity exploitation

- Outer level: retain only non-empty tiles
- Inner level: use bitmap + compact storage format

- **Packing** into compact format: performed on CPU as a preprocessing step
- **Unpacking** for OTF XMV: performed in parallel on GPU using bit magic + warp intrinsics

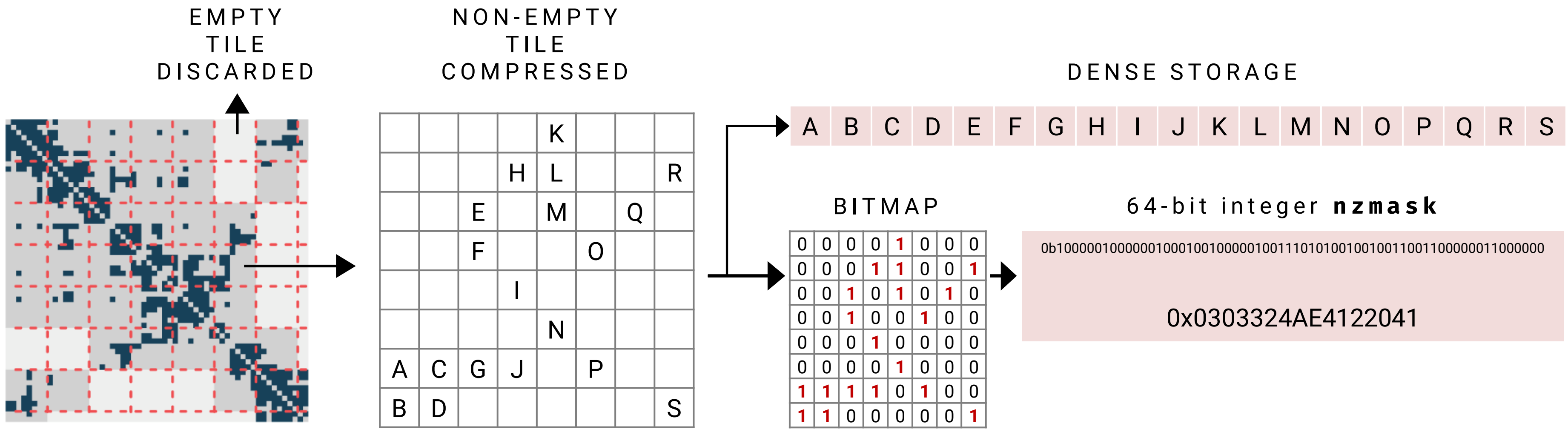


A 2-level hierarchical sparse matrix format ensures efficient memory usage

2-Level sparsity exploitation

- Outer level: retain only non-empty tiles
- Inner level: use bitmap + compact storage format

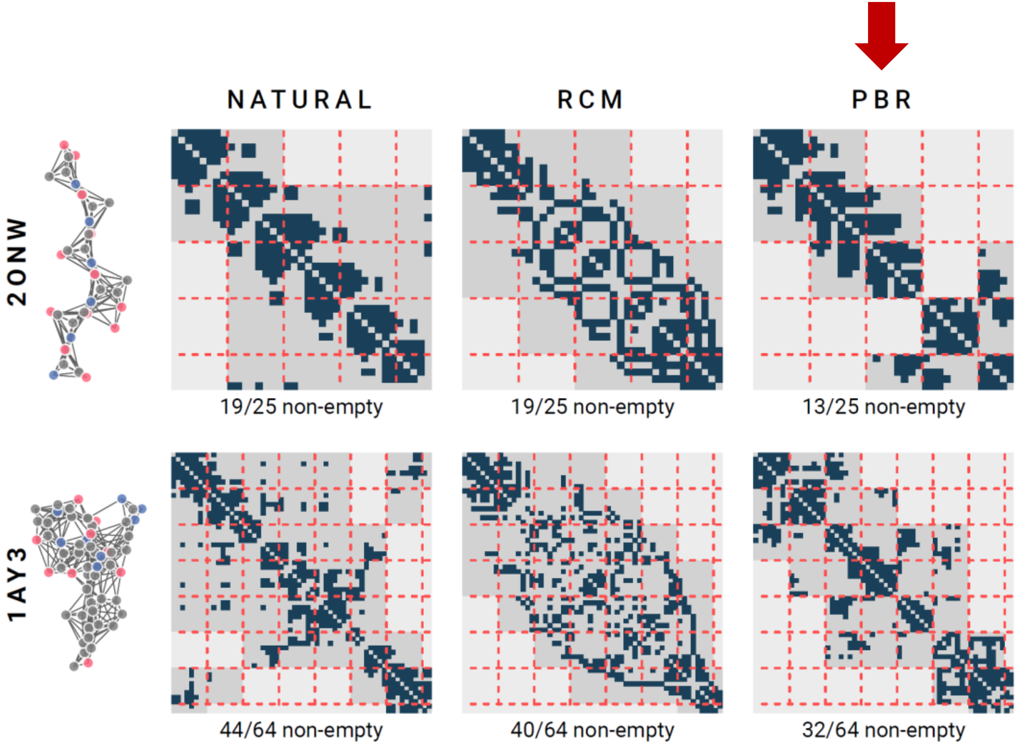
- Heuristics for dynamic code path selection:
  - If both tiles contain more than certain number of non-zero elements, treat them as dense matrices.
  - Otherwise, compute only the non-zeros.



Specialized graph reordering algorithm improves efficiency of OTF XMV and 2-level sparse format

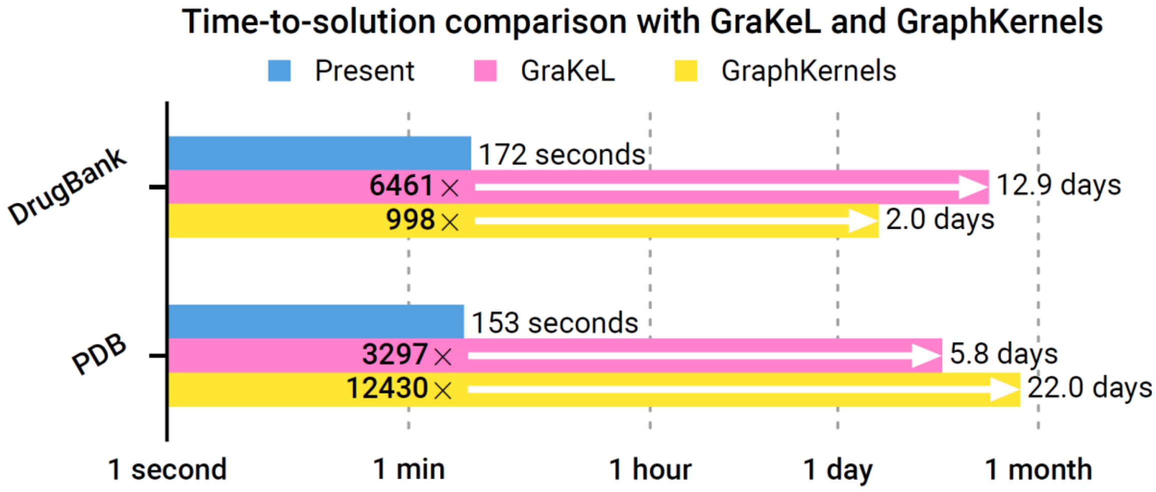
Partition-based graph reordering (PBR)

- Reduces # of non-empty sparse tiles
- Improves density of non-empty tiles
- Cost easily amortized by repeated pairwise graph kernel computations.



The On-the-Fly GPU Solver Achieves Four Orders of Magnitude Speedup Over Existing Packages

- GraKeL: Cython, multi-threading
- GraphKernels: Python, no parallelization

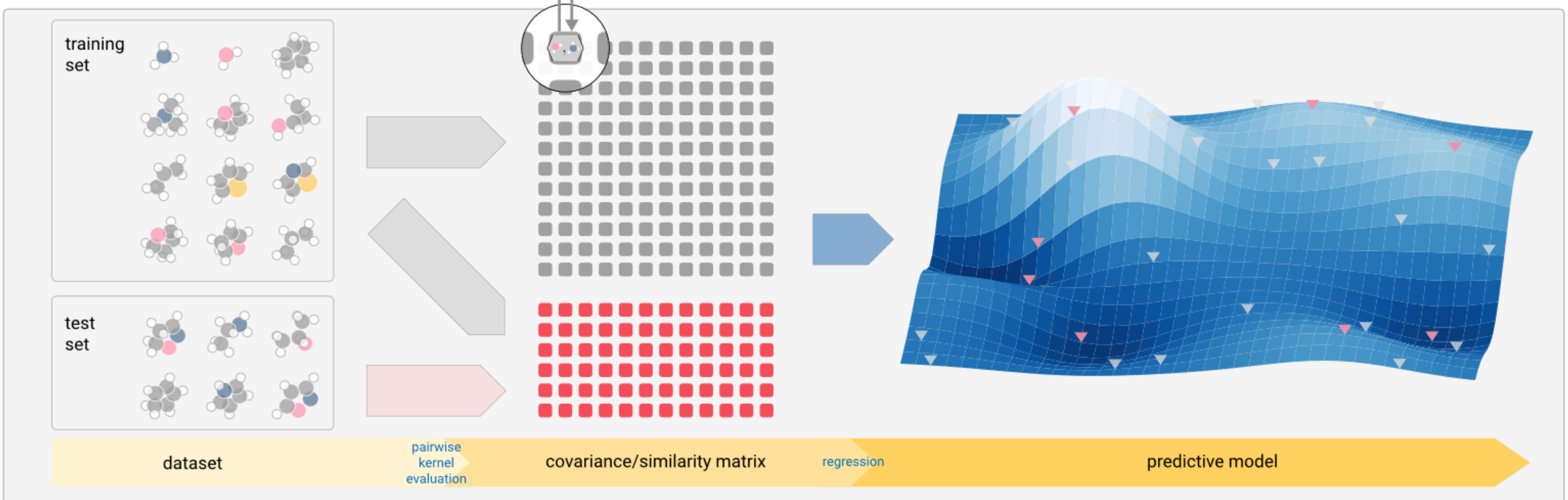


Marginalized graph kernel has found successful applications in a variety of ML tasks

- Prediction of molecular atomization energy

Tang & de Jong, J Chem Phys, 2019
Prediction of atomization energy using graph kernel and active learning
https://doi.org/10.1063/1.5078640

Graph kernel evaluation



Gaussian Process Regression on Molecules

Marginalized graph kernel enables active learning of atomization energy in orders of magnitude less time than NN

- QM7: 7165 small organic molecules consisting of H, C, N, O, S, up to 23 atoms
  - From scratch training time: N = 1000: 10 s training, 0.018 s/sample predicting, N = 2000: 40 s training, 0.034 s/sample predicting

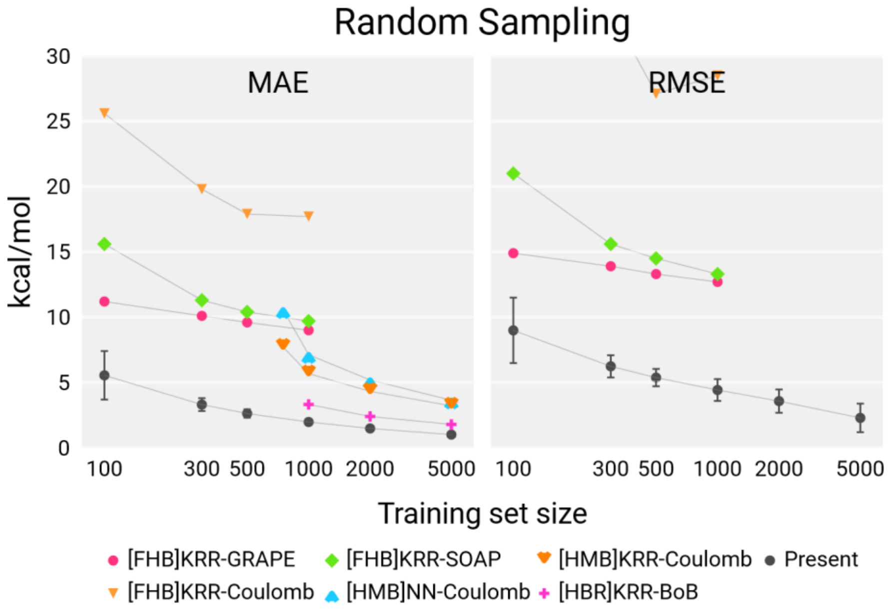
MAE: Mean Average Error

RMSE: Root-Mean Square Error

KRR: Kernel Ridge Regression

NN: Neural Network

GRAPE, SOAP, Coulomb, BoB: fingerprint algorithms



Content

SUMMARY

Summary & Acknowledgement

Graphs are useful data structures for representing scientific datasets.

The **marginalized graph kernel** is a very generic tool for machine learning on graphs.

Marginalized graph kernel can be computed very **efficiently on GPUs**

- LBNL LDRD Project "Active Learning of Ab Initio Force Fields with Applications to Large-Scale Simulations of Materials and Biophysical Systems"
- Also supported in part by the Applied Mathematics program of the DOE Office of Advanced Scientific Computing Research under Contract No. DE-AC02-05CH11231, and in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Thank You!

```
pip install graphdot
```


Tang, Selvitopi, Popovici, Buluc, IPDPS 2020: A High-Throughput Solver for Marginalized Graph Kernels on GPU.
https://arxiv.org/abs/1910.06310
Manuscript in preparation: GraphDot: A GPU-Accelerated Python Package for Graph-Based Machine Learning.