Graph Partitioning Methods for Fast Parallel Quantum Molecular Dynamics

Hristo Djidjev, Georg Hahn, Sue Mniszewski
Christian Negre, Anders Niklasson, Vivek Sandeshmuk

October 10, 2016
Talk outline

• Background and motivation of partitioning approach
  – Quantum MD background
  – Recursive polynomial expansion of Hamiltonian matrices
  – Partitioned evaluation of matrix polynomials

• Formulation of the GP problem and its application
  – CH-partitioning definition
  – Application to matrix polynomial evaluation
  – Correctness of approach

• Development of CH-partitioning algorithms

• Experimental analysis

• Conclusion
Quantum MD background

• Classical MD simulations
  – Atoms as bodies that move based on Newton’s laws of motion
  – Forces between atoms calculated using interatomic potentials
  – Positions of atoms updated in small time steps
  – Interaction models use a priori knowledge of the system
  – Cannot explain events on atomic and subatomic level

• Quantum MD simulations
  – Based on laws of quantum mechanics
  – Density functional theory (DFT) most used model
  – Second-order spectral projection (SP2) approach
    ▪ Density matrix as a function $f$ of the Hamiltonian
    ▪ Representing $f$ as a recursive polynomial expansion
Recursive polynomial matrix expansion

- Given Hamiltonian $H$, compute density matrix $D$

  $$D = \lim_{n \to \infty} f_n(f_{n-1}(\ldots f_0(H)\ldots))$$

- $f_0(X) = \alpha I - \beta X$

- $f_i(X) = \begin{cases} X^2, & \text{if } Tr[X] > N_i \\ 2X - X^2, & \text{otherwise} \end{cases}$

- The degree grows at an exponential rate, hence 20-30 iterations suffice

- Thresholding used to reduce MM complexity

  $$D = \lim_{n \to \infty} f_n t_n(\ldots f_0 t_0(H)\ldots)$$
Parallel evaluation of matrix polynomial for $D$

- Large number of time steps ($10^4$-$10^6$) – need parallelism
- Bottleneck operation $Y = X^2$ for a sparse matrix $X$
- Sparse matrix algebra
  - Works well in sequential and shared-memory environment
  - Speedup of distributed implementation goes down with the # nodes due to communication overhead
- Partitioning based approach
  - Computational overhead (total number of operations higher)
  - Reduced communication overhead
  - Scalable parallelism
Partitioned evaluation

• Model the sparsity structure of $H$ by a graph $G = G(H)$
• Partition $G$ into (overlapping) graphs $G_i$
  — core vertices of $G_1, \ldots, G_p$ form a partition of $V(G)$
  — halo vertices are neighbors of core vertices & not in the core
  — CH-partitioning (core-halo)
• Send submatrix $H_i$ of $H$ defined by $G_i$ to node $i$
• Compute polynomial $P(H_i)$ by node $i$
• Copy core elements of $P(H_i)$ to $D := P(H)$
The CH-partitioning problem

• The partitioned algorithm correctly computes during the $i$-th iteration $D(H_i)$ assuming
  – Time step is small enough so that density matrix does not change a lot in one iteration
  – Graph used for partitioning is based on $(D_{i-1}+H_i)^2$
  – Thresholding is used after each matrix computation

• CH-partitioning problem formulation:

  Given an undirected graph $G$ and $q \geq 2$, find a partition $C_1, \ldots, C_q$ of $V(G)$ with corr. halos $H_1, \ldots, H_q$ that minimizes

  $$\sum_i(|C_i| + |H_i|)^3 \quad (or, \text{alternatively, } \max_i\{|C_i| + |H_i|\}).$$
Partitioning algorithms

- Standard graph partitioning
  - Related, but different than CH-graph partitioning
  - Solvers Metis, hMetis, KaHIP

- New algorithms
  - Kernighan-Lin based
  - Simulated annealing
  - Metis+SA
Experimental setup

- Test cases motivated by physical systems

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>n</th>
<th>m</th>
<th>m/n</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>polyethylene dense crystal</td>
<td>18432</td>
<td>4112189</td>
<td>223.1</td>
<td>crystal molecule in water low threshold</td>
</tr>
<tr>
<td>2</td>
<td>polyethylene sparse crystal</td>
<td>18432</td>
<td>812343</td>
<td>44.1</td>
<td>crystal molecule in water high threshold</td>
</tr>
<tr>
<td>3</td>
<td>phenyl dendrimer</td>
<td>730</td>
<td>31147</td>
<td>42.7</td>
<td>polyphenylene branched molecule</td>
</tr>
<tr>
<td>4</td>
<td>polyalanine 189</td>
<td>31941</td>
<td>1879751</td>
<td>58.9</td>
<td>poly-alanine protein solvated in water</td>
</tr>
<tr>
<td>5</td>
<td>peptide 1aft</td>
<td>385</td>
<td>1833</td>
<td>4.76</td>
<td>ribonucleoside-diphosphate reductase protein</td>
</tr>
<tr>
<td>6</td>
<td>polyethylene chain 1024</td>
<td>12288</td>
<td>290816</td>
<td>23.7</td>
<td>chain of polymer molecule, almost 1-d</td>
</tr>
<tr>
<td>7</td>
<td>polyalanine 289</td>
<td>41185</td>
<td>1827256</td>
<td>44.4</td>
<td>large protein in water solvent</td>
</tr>
<tr>
<td>8</td>
<td>peptide trp cage</td>
<td>16863</td>
<td>176300</td>
<td>10.5</td>
<td>small protein dissolved in H$_2$O molecules</td>
</tr>
<tr>
<td>9</td>
<td>urea crystal</td>
<td>3584</td>
<td>109067</td>
<td>30.4</td>
<td>organic compound</td>
</tr>
</tbody>
</table>
Test matrices

Phenyl dendrimer system with its molecular representation (left)

2D plot representation of the Hamiltonian (middle)

Thresholded density matrix (right)
Comparison of accuracies

![Graph showing comparison of accuracies for different physical systems. The x-axis represents various physical systems: Polyeth. cr. dense, Polyeth. cr. sparse, Phenyl dendrimer, Polyalanine 189, Peptide 1aft, Polyeth. lin. chain, Polyalanine 289, Peptide trp cage, Urea crystal. The y-axis represents the sum of cubes, with values ranging from 50 to 170. The legend includes METIS, METIS+SA, hMETIS, hMETIS+SA, KaHIP, and KaHIP+SA, with different colors for each category.](image-url)
Comparison of running times
QMD running time comparison

![Graph showing running time comparison for different partition sizes.](image)
Conclusion

• New graph partitioning problem with applications in materials science and sparse matrix polynomials
  – Parts overlap
  – Objective function not directly related to edge cut

• Several implementations
  – Classical GP algorithms + SA postprocessing
  – KaHIP+SA gives best quality
  – Metis+SA best running time and best overall

• Parallel QMD implementation based on CHP runs about 10 times faster than SM based version