CS 598: Communication Cost Analysis of Algorithms
Lecture 25: Fast integral equation methods, hierarchically structured matrices

Edgar Solomonik

University of Illinois at Urbana-Champaign

November 16, 2016

Distance-limited interactions

Last lecture covered methods for direct particle interactions

- pairwise interactions among N particles with cut-off radius r_c
- uniform distribution in $N^{1/3} \times N^{1/3} \times N^{1/3}$ domain
- best algorithm attained communication cost

$$O\left(r_c\left(\frac{N}{P}\right)^{2/3} + r_c^{3/2}\sqrt{N/P}\right)$$

- the first term is r_c times the boundary of a box assigned to a processor and is dominant when r_c is small
- when r_c is big, a good approach would be to let each processor compute interactions between a unique pair of two boxes
- the total number of pairwise interactions is $\Theta(Nr_c^3)$, and given K particles there are $O(K^2)$ interactions to perform, so the two boxes would need to be of size $\Theta(\sqrt{Nr_c^3/P})$

Smooth Particle Mesh Ewald (SPME) method

Solve for long range interactions on a $m \times m \times m$ charge grid

- assume to be periodicity, which is reasonable for large systems
- Ewald summation is used to split the total potential energy

$$E = \frac{1}{2} \sum_{c \in \mathbb{Z}^3} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{|x(i) - x(j) + cN^{1/3}|}$$

into two parts (the form here is slightly simplified)

- the first part is a dampened direct summation

$$E_{\text{dir}} = \frac{1}{2} \sum_{c \in \mathbb{Z}^3} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j \text{erfc}(\beta |x(i) - x(j) + cN^{1/3}|)}{|x(i) - x(j) + cN^{1/3}|}$$

  the function $\text{erfc}(y)$ is the probability a uniform random variable with mean 0 and variance $1/2$ falls outside of the range $[-y, y]$, so pairs with sufficiently large $x(i) - x(j)$ or in distant cells can be ignored
- the second part is a convolution over the charge grid in all cells except $c = (0, 0, 0)$ contracted based on β

SPME computational structure

The forces on particles in SPME are obtained by equations that are derivatives of the energy with respect to position

- the computation of the reciprocal part is fairly simple
  - B-splines interpolate charge from nearby region of particles with cost
    $$O(m^3/P \cdot \gamma + (N/P)^{2/3} \cdot \beta + \alpha)$$
  - Q: the convolution on the $m \times m \times m$ charge grid is solved via 3D FFT, with what cost?
  - A: when $P \leq m^{5/2}$
    $$O(m^3 \log(m)/P \cdot \gamma + m^3/P \cdot \beta + \alpha)$$
  - integrating potential from grid to compute forces on particles has cost
    $$O(m^3/P \cdot \gamma + m^2/P^{2/3} \cdot \beta + \alpha)$$

- SPME performs very well when the charge density is uniform and periodic conditions are reasonable, it is also straight-forward to adjust it to handle bonded interactions

Solving 3D Poisson via multigrid

We can achieve an overall computation cost of $O(N/P)$ for MD, when $m^3 \approx N$ by using multigrid

- grid construction and charge interpolation is different but possible
- multigrid V-cycle with grid $m_i \times m_i \times m_i$ has the following costs
  - $O(1)$ smoothing iterations per level have cost
    $$O(m_i^3/P \cdot \gamma + m_i^2/P^{2/3} \cdot \beta + \alpha)$$
  - interpolation between grids and restriction asymptotically cost the same as smoothing
  - Q: why? and does this still hold for smoothed algebraic multigrid?
  - A: generally (nearly-)adjacent nodes in the mesh are combined, smoothing propagates information also only from adjacent nodes
- assuming $m_0 = m$ and $m_i = m_{i-1}/2$, need $O(\log(P))$ levels before subgrid can be solved by one processor, so overall cost is
  $$O(m^3/P \cdot \gamma + m^2/P^{2/3} \cdot \beta + \log(P) \cdot \alpha)$$
- so SPME with FFT may be slightly more synchronization-efficient

Short pause

Barnes-Hut

If the particle distribution is nonuniform, a regular grid is unsuitable

- main alternative is tree-based decomposition
- subdivide space recursively until each cell contains $O(k)$ particles
  - in 1D, obtain binary-tree
  - in 2D, obtain quad-tree
  - in 3D, obtain oct-tree
- compute a centered mass/charge for each tree node or r terms of a Taylor series for higher accuracy
- calculate forces between far-away particles in far-away cells, based on interaction with particle and a mass/charge at a higher-level tree node

Barnes-Hut

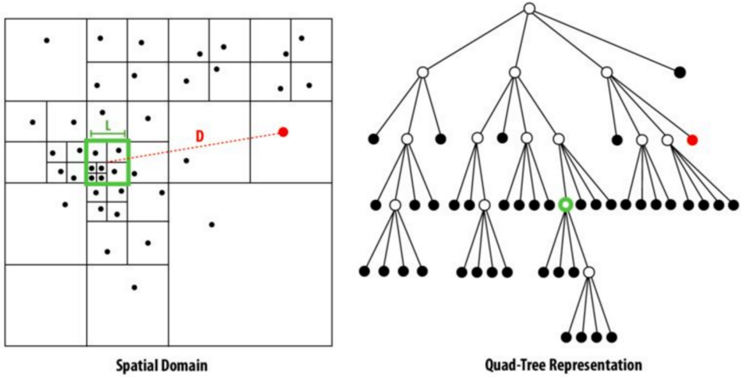


Diagram taken from course webpage of Mowry and Railing (CMU)

Hierarchical matrices

$\mathcal{H}$-matrices or HSS (hierarchically semi-separable matrices), introduced by Hackbusch, are an algebraic representation of a Barnes-Hut-like algorithm

- let $A \in N \times N$ encode the desired interactions
- given a binary tree partitioning, we split up

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

  if the partitioning is balanced among particles A_{ij} are $n/2 \times n/2$
- Q: how would a quad tree partitioning look like?
- A: a quad tree partitioning would give

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

- in an $\mathcal{H}$-matrix each A_{ii} is partitioned recursively and each off-diagonal block A_{ij} is rank r

Computation with $\mathcal{H}$-matrices

A step in Barnes-Hut simulation looks like a matrix-vector product with a $\mathcal{H}$-matrix, $y = Ax$

- consider the 1D case (binary tree), 2D and 3D are similar
- to update first partition of particles, we perform $y_1 = A_{11}x_1 + A_{12}x_2$
- thus for the particles in the second partition we can perform $U_1 V_2^T x_2$, where $A_{12} = U_1 V_2^T$ and $U_1, V_2^T \in \mathbb{R}^{n \times r}$
- if we just have a single center of mass $r = 1$, the off-diagonal blocks are rank 1
- Q: how much computation is required for $y = Ax$ if r is a constant and there are $O(\log(n))$ levels?
- A: $T(n) = 2T(n/2) + O(rn) = O(rn\log(n))$, this is also the amount of storage needed for A
- its also possible to define other operations including matrix-matrix multiplication and LU factorization with $\mathcal{H}$-matrices

Communication cost with $\mathcal{H}$-matrices

Lets consider parallel $\mathcal{H}$-matrix-vector multiplication

- lets assume r is small
- Q: if we use a 1D blocking for U_1 and V_2, how much communication is required for $U_1 V_2^T x_2$?
- A: $O(r)$
- thus, given a proper blocking, the BSP complexity is

$$\begin{aligned} T(n,P,r) &= T(n/2,P/2,r) + O(nr/P\cdot\gamma + r\cdot\beta + \alpha) \\ &= O(nr\log(n)/P\cdot\gamma + r\log(P)\cdot\beta + \log(P)\cdot\alpha) \end{aligned}$$

- construction of the $\mathcal{H}$-matrix (oct-tree) costs somewhat more
- when partitions are not load balanced, processors should be partitioned accordingly, and height of tree may increase

Fast multipole method (FMM)

The FMM algorithm obtains linear complexity for integral equations

- there are many derivations of FMM for different types of equations
- the first, by Greengard and Rokhlin was for 2D electrostatics
- like in Barnes-Hut a tree of particles is defined, but in FMM we interact non-leaf nodes in the tree, so that every particle interacts with $O(1)$ tree nodes
- for each tree node a multipole (inner) and Taylor (outer) expansion is defined consisting of $r = O(\log(1/\epsilon))$ terms for accuracy ϵ
    - thus, error is controlled explicitly by the size of the expansion
    - a multipole expansion is a special type of Taylor expansion
- *transformation* operators are defined to 'shift' multipole and Taylor expansions, and to convert between the two

FMM algorithm

The computation in FMM proceeds as follows

1. perform interactions among local particles
2. upward pass – generate multipole expansion for every tree node starting from leaves
3. downward pass – generate local expansion for every tree node starting from root

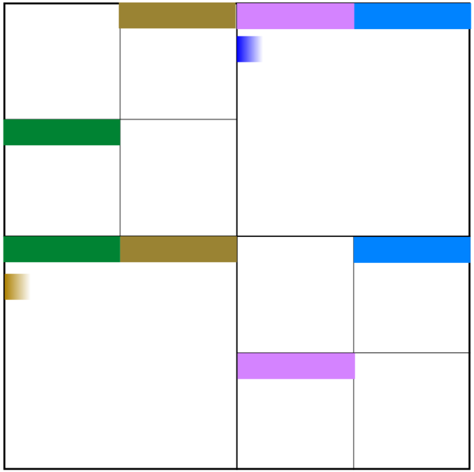
FMM as $\mathcal{H}^2$-matrices

$\mathcal{H}^2$-matrices provide an alternative closely-related way for solving integral equations with linear complexity

- $\mathcal{H}^2$-matrices may be seen as a specialization of $\mathcal{H}$-matrices
- we represent each off-diagonal block at the lth level as $A_{ij}^{(l)} = W_i^{(l)} K_{ij}^{(l)} (V_j^{(l)})^T$
- if each $K_{ij}^{(l)} \in \mathbb{R}^{r\times r}$ this is just an $\mathcal{H}$-matrix
- what makes $\mathcal{H}^2$-matrices special is a nested structure of the basis $W_i^{(l)}$ and $V_j^{(l)}$
- in particular *transformation matrices* $R_i^{(l)}$, $T_j^{(l)}$ define these with respect to the finer level $l-1$
- if branch factor is s and matrices are appropriately indexed then

$$W_i^{(l)} = \begin{bmatrix} W_{(i-1)s+1}^{(l-1)} \\ \vdots \\ W_{is}^{(l-1)} \end{bmatrix} R_i^{(l)}, \qquad V_j^{(l)} = \begin{bmatrix} V_{(j-1)s+1}^{(l-1)} \\ \vdots \\ V_{js}^{(l-1)} \end{bmatrix} T_j^{(l)}$$

Depiction of $\mathcal{H}^2$-matrices



Depicted are $(V_{js}^{(l-1)})^{\mathsf{T}}$ and $(T_j^{(l)})^{\mathsf{T}}$ and not the other parts of the updates

Computation with $\mathcal{H}^2$-matrices

Matrix-vector multiplication with a $\mathcal{H}^2$-matrix can be dine in $O(nr)$ operations

- requires forward transformation, multiplication, and backward transformation
- can compute $(V_j^{(l)})^\mathsf{T} x_j^{(l)} = (T_j^{(l)})^\mathsf{T} \sum_{k=1}^{s} (V_{(j-1)s+k}^{(l-1)})^\mathsf{T} x_j^{(l-1)}$
- requires $O(r^2)$ work
- typically $s = 1$ and $r = \log(1/\epsilon)$
- at leaves need to perform $O(rn)$ work
- naive parallelization of tree would yield BSP complexity (assuming small r)

$$T(n, P, r) = O(nr/P \cdot \gamma + r\log(P) \cdot \beta + \log(P) \cdot \alpha)$$

- may be possible to shed $\log(P)$ factor