Efficient Massively Parallel Methods for Dynamic Programming

Benjamin Moseley
Washington University in St. Louis

Sungjin Im and Xiaorui Sun
STOC 2017
Revolution in Scalability
Breakthroughs Driven by Practitioners
Massively Parallel Computation
Massively Parallel Computation

Local Data

Computation
Models

- Models first motivated by MapReduce
- Refined and parameterized
  - Capture most modern massively parallel environments

Dean-Ghemawat OSDI 2004,
Feldman-Muthukrishnan-Sidiropoulos-Stein-Svitkina SODA 2008 (TALG 2010),
Karloff-Suri-Vassilvitskii SODA 2010,
Goodrich-Sitchinava-Zhang ISAAC 2011,
Pietracaprina-Pucci-Riondato-Silvestri-Upfal ICS 2012
Goel-Mungagala ArXiv 2012,
Beame-Joutris-Suciu PODS 2013,
Andoni-Nokolov-Onak-Yaroslavtev STOC 2014,
Roughgarden-Vassilvitskii SPAA 2016
Unifying Models

- $O(n)$ input size
- $O(m)$ number of machines
- $\tilde{O}(n/m)$ memory on the machines
- Minimize rounds
Unifying Models

- $O(n)$  input size
- $O(m)$  number of machines
- $\tilde{O}(n/m)$ memory on the machines
  - $\tilde{O}(n^{1+\frac{1}{2}\epsilon}/m)$ ‘extra’ memory setting
- Minimize rounds
Algorithmic Challenges

Local Data
Dynamic Programming
Principled Framework

- Identify two key properties enabling simulation
  - Monotonicity
  - Decomposability
- $(1 + \epsilon)$-approximations in $O\left(\frac{1}{\delta}\right)$ rounds using $\tilde{O}(n)$ aggregate memory, $O(n^\delta)$ memory per machine for any constants $\delta, \epsilon > 0$
  - Optimal binary search tree
  - Longest increasing subsequence
  - Weighted interval selection
Weighted Interval Selection

- Collection of intervals $I_i = (s_i, e_i)$ with weight (profit) $w_i$
- Select a maximum weight independent set of intervals

- Dynamic program
  - Sort intervals by starting points in increasing order
  - $A(i)$ optimal solution of $I_i, I_{i+1}, \ldots, I_n$
  - $A(i) = \max\{A(i + 1), w_i + A(j)\}$ where $j = \text{argmin}_{j', e_i < s_{j'}'}$
Distributed Algorithm

- $m$ machines

- Distribute intervals to machines in sorted order

- $n/m$ intervals per machine
Distributed Algorithm

• Compute subproblems locally, but which ones?
  
  • \( B(i, j) \) start no earlier than \( s_i \) and end by \( s_j \)

• Combine on a single machine

• Super-linear space!
Logarithmic Rounds using Monotonicity

• Any subproblem has less optimum value for a maximization objective

• \( B(i, j) \) satisfies monotonicity, \( B(i', j') \geq B(i, j) \) for all \( i' \leq i, j' \geq j \)
Swap: A use of Monotonicity

• Define a new recurrence \( C(i, w) = \min_{j' \mid B(i, j') \geq w} j' \)

• Increasing due to monotonicity

• Set \( C(i, w) = \min j_2 \) over all \( w_1, w_2 \) where \( w_1 + w_2 = w \)

\( C(i, w_1) = j_1 \) and \( C(j_1, w_2) = j_2 \)
Sketching the Recurrence

- Approximately sketch the recurrence

  - Compute $C(i, w)$ where $w$ is of the form $(1 + \frac{\epsilon}{10 \log n})^k$ to ensure polylogarithmic space

- Need a new approximate recurrence $C'(i, w)$

- Recursively compute by considering $w_1, w_2$ where $w \simeq w_1 + w_2$

- Set $C'(i, w) = \min j_2$ such that $C'(i, w_1) = j_1$ and $C'(j_1, w_2) = j_2$

- Loose $(1 + \frac{\epsilon}{10 \log n})$ factor each iteration however for small $k$

  $$(1 + \frac{\epsilon}{10 \log n})^k \leq 1 + \epsilon$$
Computing the Sketch

• Initialize $C'(i, w)$ by considering only intervals on the machine $i$ is assigned to

• In the $k$th round $C'(i, w)$ will contain the optimal value amongst intervals on machines up to $2^k$ away
Computing the Sketch

• Updating $C'(i, w)$

• Consider all pairs $w_1, w_2$ where $w_1 + w_2 \simeq w$

• When $C'(i, w_1) = j_1$ send the value $C'(j_1, w_2) = j_2$ to the machine $i$ is stored on
  
  • Computed at the beginning of the round

• Update $C''(i, w)$ to be the minimum such $j_2$
Generalizing the Idea

• Framework extends to other problems

• Combining with decomposability requires new ideas
Future Work

• General methods

• Lower bounds
  [Sarma-Afrati-Salihoglu-Ullman; Fish-Kun-Lelkes-Reyzin-Turaan; Jacob-Lieber-Sitchinava; Beame-Koutris-Suciu; Roughgarden-Vassilvitskii-Wang]
Thank you!

Questions?