MedPart: A Multi-Level Evolutionary Differentiable Hypergraph Partitioner

Rongjian Liang, Anthony Agnesina, Mark Ren

NVIDIA
OUTLINE

• Motivations And Contributions
• Problem Formulation
• MedPart
  • Spectral Coarsening and Multi-Level Optimization
  • Evolutionary Differentiable Hypergraph Partitioning
  • Acceleration By Deep Graph Learning Toolkits on GPUs
• Experimental Validation
• Conclusions And Future Directions
Motivations

Limitations of SOTA partitioners call for new hypergraph partitioning algorithms

- Hypergraph partitioning is a foundational problem in EDA
- State-of-the-art partitioners follow a multi-level paradigm, progressively coarsening hypergraphs to explore a vast solution space efficiently, but they
  - Overlooking global structural information during coarsening
  - Rely on refinement heuristics during local improvement
- SpecPart refines solutions by spectral information, but relying on good initial solutions
Contributions
A novel analytical optimization framework for better partitioning quality

• A multi-level evolutionary differentiable hypergraph partitioner named MedPart

How to improve hypergraph partitioning quality?

• Feature 1: A fast spectral hypergraph coarsening algorithm (capturing global info)
• Feature 2: An evolutionary differentiable algorithm that integrates genetic algorithm with gradient descent for optimization at each coarsening level (analytical optimization rather than heuristics)
• Feature 3: Accelerate our evolutionary differentiable optimizer with deep graph learning toolkits on GPUs by analogy between hypergraph partitioning and deep graph learning
OUTLINE

• Motivations And Contributions

• Problem Formulation

• MedPart
  • Spectral Coarsening and Multi-Level Optimization
  • Evolutionary Differentiable Hypergraph Partitioning
  • Acceleration By Deep Graph Learning Toolkits on GPUs

• Experimental Validation

• Conclusions And Future Directions
Problem Formulation

Divide a hypergraph into multiple nearly equal-sized parts while minimizing cut edges

A hypergraph $H$ is defined as a pair $H = (V, E)$ where $V$ represents the set of vertices $v \in V$ with associated weight $w_v$, and $E$ represents the set of hyperedges where an hyperedge $e \in E$ is a subset of $V$ with associated weight $w_e$. Given a positive integer $k \geq 2$ and a positive real number $\epsilon \leq \frac{1}{k}$, letting $W = \sum_{v \in V} w_v$, the $k$-way balanced hypergraph partitioning problem can be mathematically formulated as:

$$\min_{S=\{V_1, V_2, \ldots, V_k\}} \text{cutsize}_H(S) = \sum_{\{e \mid e \not\in V_i, \forall i\}} w_e \quad (1)$$

subject to $\bigcup_{i=1}^{k} V_i = V$ and $V_i \cap V_j = \emptyset$, $0 \leq i, j \leq k$ \quad (2)

$$\left(\frac{1}{k} - \epsilon\right) W \leq \sum_{v \in V_i} w_v \leq \left(\frac{1}{k} + \epsilon\right) W, \quad 0 \leq i \leq k, \quad (3)$$

where Equation (2) ensures that $S$ is a $k$-way disjoint partitioning solution of $V$, and $\epsilon$ is the allowed imbalance between partitions (Equation (3)). We say that $S$ is an $\epsilon$-balanced partitioning solution.

Minimize cut size

Non-overlap between partition blocks

Balanced partition block sizes
OUTLINE

• Motivations And Contributions
• Problem Formulation
• MedPart
  • Spectral Coarsening and Multi-Level Optimization
    • Evolutionary Differentiable Hypergraph Partitioning
    • Acceleration By Deep Graph Learning Toolkits on GPUs
• Experimental Validation
• Conclusions And Future Directions
Spectral Coarsening and Multi-Level Optimization

- **Phase 1:** **Graph coarsening** (top-down)
  - Hypergraph -> Clique expansion graph
  - Progressive graph coarsening on the clique expansion graph by **graph signal processing-based fast spectral coarsening technique** [1]
  - Construct the projection matrices

- **Phase 2:** **Coarse-to-grain partitioning** (bottom-up)
  - Enumeration or evolutionary differentiable algorithm at each level
  - Coarser-level solutions are mapped to solutions at finer using the projection matrices and act as starting points

---

OUTLINE

• Motivations And Contributions
• Problem Formulation
• MedPart
  • Spectral Coarsening and Multi-Level Optimization
  • Evolutionary Differentiable Hypergraph Partitioning
    • Acceleration By Deep Graph Learning Toolkits on GPUs
• Experimental Validation
• Conclusions And Future Directions
Evolutionary Differentiable Hypergraph Partitioning

Gradient descent-based analytical optimization + Genetic algorithm for initialization

- Generate M offsprings by crossover and mutation
- M gradient descent (GD) trials runs in parallel to optimize the M offsprings
- Update population with the optimized solutions from GD

```
Algorithm 2: Evolutionary Differentiable Hypergraph Partitioning Algorithm

Input: 1: number of generations; M: population size; Th: stagnation threshold
1: number of GD steps; T: checkpoint steps
2: Xu := \{x_0^0, x_0^1, \ldots, x_0^n\}: initial population
Output: x*: best partitioning solution
1: /* Evaluation */
2: scores(X_0) = EvalFitness(X_0)
3: /* 1 generations */
4: for i := 1 to do
5: /* Generate offspring population by crossover and mutation */
6: \{c_1^0, \ldots, c_M^0\} := GenOffspring(X_{i-1}, scores(X_{i-1}))
7: /* Evaluation */
8: scores(c_1^0, \ldots, c_M^0) = EvalFitness(c_1^0, \ldots, c_M^0)
9: /* in parallel by batching */
10: for m := 1 to M do
11: /* GD epoch */
12: Initialize the best solution for the current GD run: c_M^m := c_m^0.
13: scores(c_M^m) := scores(c_m^0)
14: Select hyperparameters \pi_m
15: Initialize continuous solution: \hat{c}_m^0 := Relax(c_m^0)
16: for s := 1 to S do
17: GD update of \hat{c}_m^s with \pi_m
18: if (s mod T == 0) or (s == S) then
19: c_m^s := Discretize(\hat{c}_m^s)
20: scores(c_m^s) := EvalFitness(c_m^s)
21: if scores(c_m^s) better than scores(c_M^m) then
22: C_M^m := c_m^s
23: scores(c_M^m) := scores(c_m^s)
24: end
25: end
26: end
27: /* Gather best solutions from GD outcomes */
28: C_i^* := \{c_1^*, c_2^*, \ldots, c_M^*\}
29: scores(C_i^*) := \{scores(c_1^*), \ldots, scores(c_M^*)\}
30: /* Update population with deterministic crowding */
31: X_i := UpdatePopulation(X_{i-1}, scores(X_{i-1}), C_i^*, scores(C_i^*))
32: /* Early stop criterion */
33: if the best fitness score does not improve for over Th generations then
34: x* := best solution from X_i
35: return x*
36: end
37: end
38: return the best solution x* among X_i
```
OUTLINE

• Motivations And Contributions
• Problem Formulation
• MedPart
  • Spectral Coarsening and Multi-Level Optimization
  • Evolutionary Differentiable Hypergraph Partitioning
  • Acceleration By Deep Graph Learning Toolkits on GPUs
• Experimental Validation
• Conclusions And Future Directions
Acceleration By Deep Graph Learning Toolkits on GPUs

Leverage analogy between hypergraph partitioning and deep graph learning

Figure 3: Batch cut size evaluation and optimization on the Hypergraph-Node Relationship graph. A batch of candidate assignments for each node is aggregated into the hyperedges to calculate objectives. (a) Batch cut size evaluation with discrete node to partition assignments. (b) Batch differentiable cut size optimization with soft probabilistic node to partition assignments. By analogy with deep graph learning, both cut-size evaluation and optimization can be accelerated with deep graph learning toolkits on GPUs.

Cut size evaluation ↔ Forward message passing in graph neural network
Cut size minimization ↔ Backward gradient propagation in graph neural network
OUTLINE

• Motivations And Contributions
• Problem Formulation
• MedPart
  • Spectral Coarsening and Multi-Level Optimization
  • Evolutionary Differentiable Hypergraph Partitioning
  • Acceleration By Deep Graph Learning Toolkits on GPUs
• Experimental Validation
• Conclusions And Future Directions
Results on ISPD98 VLSI Circuit Benchmark Suite
Small gap to the best-published results

Table 1: Statistics of ISPD98 VLSI circuit benchmark suite and cut sizes of different approaches. SOTA represents the best-published cut sizes summarized in [9]. Spec denotes the Specpart result presented in [6], which is obtained by employing SpecPart to enhance partitioning solutions generated by hMETIS and/or KaHyPar. hMETIS_5 signifies the best cut size obtained from running hMETIS 5 times with different random seeds (provided in [9]). MedPart and MedPart_{h5} represent the cut sizes resulting from running MedPart once from scratch and using MedPart to refine the solutions from hMETIS_5, respectively. The best and the second-best results among all the methods are highlighted in red and blue, respectively.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>[V]</th>
<th>[E]</th>
<th>SOTA</th>
<th>Spec</th>
<th>hMETIS_5</th>
<th>MedPart</th>
<th>MedPart_{h5}</th>
<th>SOTA</th>
<th>Spec</th>
<th>hMETIS_5</th>
<th>MedPart</th>
<th>MedPart_{h5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM01</td>
<td>12752</td>
<td>14111</td>
<td>200</td>
<td>202</td>
<td>213</td>
<td>202</td>
<td>205</td>
<td>166</td>
<td>171</td>
<td>190</td>
<td>166</td>
<td>166</td>
</tr>
<tr>
<td>IBM02</td>
<td>19601</td>
<td>19584</td>
<td>307</td>
<td>336</td>
<td>339</td>
<td>332</td>
<td>339</td>
<td>262</td>
<td>262</td>
<td>262</td>
<td>264</td>
<td>262</td>
</tr>
<tr>
<td>IBM03</td>
<td>23156</td>
<td>27401</td>
<td>351</td>
<td>959</td>
<td>972</td>
<td>955</td>
<td>957</td>
<td>950</td>
<td>952</td>
<td>960</td>
<td>955</td>
<td>954</td>
</tr>
<tr>
<td>IBM04</td>
<td>27507</td>
<td>31970</td>
<td>573</td>
<td>593</td>
<td>617</td>
<td>583</td>
<td>584</td>
<td>388</td>
<td>388</td>
<td>388</td>
<td>389</td>
<td>388</td>
</tr>
<tr>
<td>IBM05</td>
<td>29347</td>
<td>28446</td>
<td>1706</td>
<td>1720</td>
<td>1744</td>
<td>1748</td>
<td>1744</td>
<td>1645</td>
<td>1688</td>
<td>1733</td>
<td>1675</td>
<td>1668</td>
</tr>
<tr>
<td>IBM06</td>
<td>32498</td>
<td>34826</td>
<td>962</td>
<td>963</td>
<td>1037</td>
<td>1000</td>
<td>1012</td>
<td>728</td>
<td>733</td>
<td>760</td>
<td>788</td>
<td>760</td>
</tr>
<tr>
<td>IBM07</td>
<td>45926</td>
<td>48117</td>
<td>578</td>
<td>935</td>
<td>975</td>
<td>913</td>
<td>916</td>
<td>760</td>
<td>760</td>
<td>796</td>
<td>773</td>
<td>772</td>
</tr>
<tr>
<td>IBM08</td>
<td>51309</td>
<td>50513</td>
<td>1140</td>
<td>1146</td>
<td>1146</td>
<td>1158</td>
<td>1146</td>
<td>1120</td>
<td>1140</td>
<td>1145</td>
<td>1131</td>
<td>1135</td>
</tr>
<tr>
<td>IBM09</td>
<td>53395</td>
<td>60062</td>
<td>620</td>
<td>620</td>
<td>637</td>
<td>625</td>
<td>623</td>
<td>519</td>
<td>519</td>
<td>535</td>
<td>520</td>
<td>520</td>
</tr>
<tr>
<td>IBM10</td>
<td>69429</td>
<td>75196</td>
<td>1253</td>
<td>1318</td>
<td>1313</td>
<td>1327</td>
<td>1295</td>
<td>1244</td>
<td>1261</td>
<td>1284</td>
<td>1259</td>
<td>1257</td>
</tr>
<tr>
<td>IBM11</td>
<td>70558</td>
<td>81454</td>
<td>1051</td>
<td>1062</td>
<td>1114</td>
<td>1069</td>
<td>1067</td>
<td>763</td>
<td>764</td>
<td>782</td>
<td>774</td>
<td>765</td>
</tr>
<tr>
<td>IBM12</td>
<td>71076</td>
<td>77240</td>
<td>919</td>
<td>1920</td>
<td>1982</td>
<td>1955</td>
<td>1949</td>
<td>1841</td>
<td>1842</td>
<td>1940</td>
<td>1914</td>
<td>1872</td>
</tr>
<tr>
<td>IBM13</td>
<td>84199</td>
<td>96666</td>
<td>873</td>
<td>871</td>
<td>978</td>
<td>850</td>
<td>850</td>
<td>785</td>
<td>785</td>
<td>785</td>
<td>785</td>
<td>785</td>
</tr>
<tr>
<td>IBM14</td>
<td>147605</td>
<td>152772</td>
<td>842</td>
<td>1589</td>
<td>1967</td>
<td>1876</td>
<td>1884</td>
<td>1509</td>
<td>1768</td>
<td>1665</td>
<td>1639</td>
<td>1608</td>
</tr>
<tr>
<td>IBM15</td>
<td>161570</td>
<td>186608</td>
<td>2730</td>
<td>2741</td>
<td>2886</td>
<td>2855</td>
<td>2855</td>
<td>2135</td>
<td>2235</td>
<td>2262</td>
<td>2169</td>
<td>2166</td>
</tr>
<tr>
<td>IBM16</td>
<td>183484</td>
<td>190948</td>
<td>1827</td>
<td>1915</td>
<td>2059</td>
<td>1972</td>
<td>2095</td>
<td>1619</td>
<td>1619</td>
<td>1708</td>
<td>1645</td>
<td>1651</td>
</tr>
<tr>
<td>IBM17</td>
<td>185495</td>
<td>189581</td>
<td>2270</td>
<td>2354</td>
<td>2520</td>
<td>2336</td>
<td>2338</td>
<td>1989</td>
<td>1989</td>
<td>2300</td>
<td>2024</td>
<td>2028</td>
</tr>
<tr>
<td>IBM18</td>
<td>210613</td>
<td>201929</td>
<td>1521</td>
<td>1535</td>
<td>1587</td>
<td>1587</td>
<td>1587</td>
<td>1520</td>
<td>1537</td>
<td>1550</td>
<td>1829</td>
<td>1550</td>
</tr>
</tbody>
</table>

Average gap to SOTA:
- Best published results: 0%
- Leading partitioners: 2.30%
- MedPart from scratch: 5.00%
- MedPart refinement: 3.70%
- MedPart refinement from scratch: 5.30%
- SOTA: 2.10%
- Best: 3.40%
- Leading: 1.80%
Results on Titan23 Benchmark Suite
Up to 30% smaller cut size than best-published results

Table 2: Statistics of Titan23 benchmark suite and cut sizes of different approaches. SOTA represents the best-published cut sizes. $Spec_{\text{best}}$ denotes the SpectPart cut size presented in [6], which is obtained by employing SpecPart to enhance partitioning solutions generated by running hMETIS 20 times. $hMETIS_5$ signifies the best cut size obtained from running hMETIS 5 times (provided in [9]). MedPart and MedPart$_{h5}$ represent the cut sizes resulting from running MedPart once from scratch and refining the solutions from $hMETIS_5$, respectively. We utilize underlining to emphasize the cut sizes achieved by MedPart and MedPart$_{h5}$ by that outperform the SOTA.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Statistics</th>
<th>$\epsilon = 2%$</th>
<th>$\epsilon = 20%$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>V</td>
<td>$</td>
</tr>
<tr>
<td>spart1_t_core</td>
<td>91976</td>
<td>92827</td>
<td>977</td>
</tr>
<tr>
<td>t70</td>
<td>19920</td>
<td>22305</td>
<td>270</td>
</tr>
<tr>
<td>stereo_vision</td>
<td>94050</td>
<td>127085</td>
<td>169</td>
</tr>
<tr>
<td>des90</td>
<td>111221</td>
<td>139557</td>
<td>372</td>
</tr>
<tr>
<td>SLAM_sphereic</td>
<td>113115</td>
<td>142408</td>
<td>1061</td>
</tr>
<tr>
<td>cholesky_mc</td>
<td>113250</td>
<td>144948</td>
<td>285</td>
</tr>
<tr>
<td>segmentation</td>
<td>138295</td>
<td>179651</td>
<td>118</td>
</tr>
<tr>
<td>bitonic_mesh</td>
<td>192064</td>
<td>235328</td>
<td>584</td>
</tr>
<tr>
<td>darts</td>
<td>202394</td>
<td>223301</td>
<td>788</td>
</tr>
<tr>
<td>openCV</td>
<td>217433</td>
<td>284168</td>
<td>481</td>
</tr>
<tr>
<td>slap_qrd</td>
<td>240240</td>
<td>290123</td>
<td>398</td>
</tr>
<tr>
<td>mines</td>
<td>261388</td>
<td>320540</td>
<td>215</td>
</tr>
<tr>
<td>cholesky_bdti</td>
<td>266422</td>
<td>342688</td>
<td>1156</td>
</tr>
<tr>
<td>denoise</td>
<td>275638</td>
<td>356848</td>
<td>416</td>
</tr>
<tr>
<td>spart2_t_core</td>
<td>300109</td>
<td>306263</td>
<td>1227</td>
</tr>
<tr>
<td>gsm_switch</td>
<td>493260</td>
<td>507821</td>
<td>1827</td>
</tr>
<tr>
<td>mes_noc</td>
<td>547344</td>
<td>577664</td>
<td>634</td>
</tr>
<tr>
<td>LU230</td>
<td>574372</td>
<td>669477</td>
<td>3273</td>
</tr>
<tr>
<td>LU_Network</td>
<td>635456</td>
<td>769999</td>
<td>525</td>
</tr>
<tr>
<td>spart1_t_chip2</td>
<td>820886</td>
<td>821274</td>
<td>899</td>
</tr>
<tr>
<td>directr</td>
<td>931275</td>
<td>1374742</td>
<td>574</td>
</tr>
<tr>
<td>bitcoin_miner</td>
<td>1089284</td>
<td>1448151</td>
<td>1297</td>
</tr>
<tr>
<td>Avg gap to SOTA</td>
<td>0%</td>
<td>1.9%</td>
<td>31.0%</td>
</tr>
</tbody>
</table>
Runtime

Figure 4: MedPart runtime on hypergraphs with different #edges.

Linear runtime scaling, but still large room to improve
Ablation Studies

Figure 5: Impact of multi-level optimization on MedPart. The experiments are conducted on the top 15 benchmarks from the Titan23 benchmark suite, with ε set to 10%.

Multi-level paradigm is helpful

Figure 6: Cut sizes from MedPart and hMETIS on (a) sparcT1_core and (b) bitonic_mesh, each across 5 runs with different random seeds.

Stable performance of MedPart
OUTLINE

• Motivations And Contributions
• Problem Formulation
• MedPart
  • Spectral Coarsening and Multi-Level Optimization
  • Evolutionary Differentiable Hypergraph Partitioning
  • Acceleration By Deep Graph Learning Toolkits on GPUs
• Experimental Validation
• Conclusions And Future Directions
Conclusions

• We develop MedPart, a novel multi-level evolutionary differentiable hypergraph partitioning framework

• MedPart consistently outperforms the leading partitioner hMETIS on public benchmarks and achieves up to a 30% improvement in cut size compared to the best published solutions for some benchmarks
Future Directions

• Further improve runtime efficiency and quality of solutions
• Scale to hypergraphs with 100M vertices/edges
• Apply to other partitioning problem formulations, e.g., timing-driven netlist partitioning
Looking forwards to your valuable feedback/comments!
(email: rliang@nvidia.com)