MedPart: A Multi-Level Evolutionary Differentiable Hypergraph Partitioner

Rongjian Liang, Anthony Agnesina, Mark Ren

NVIDIA

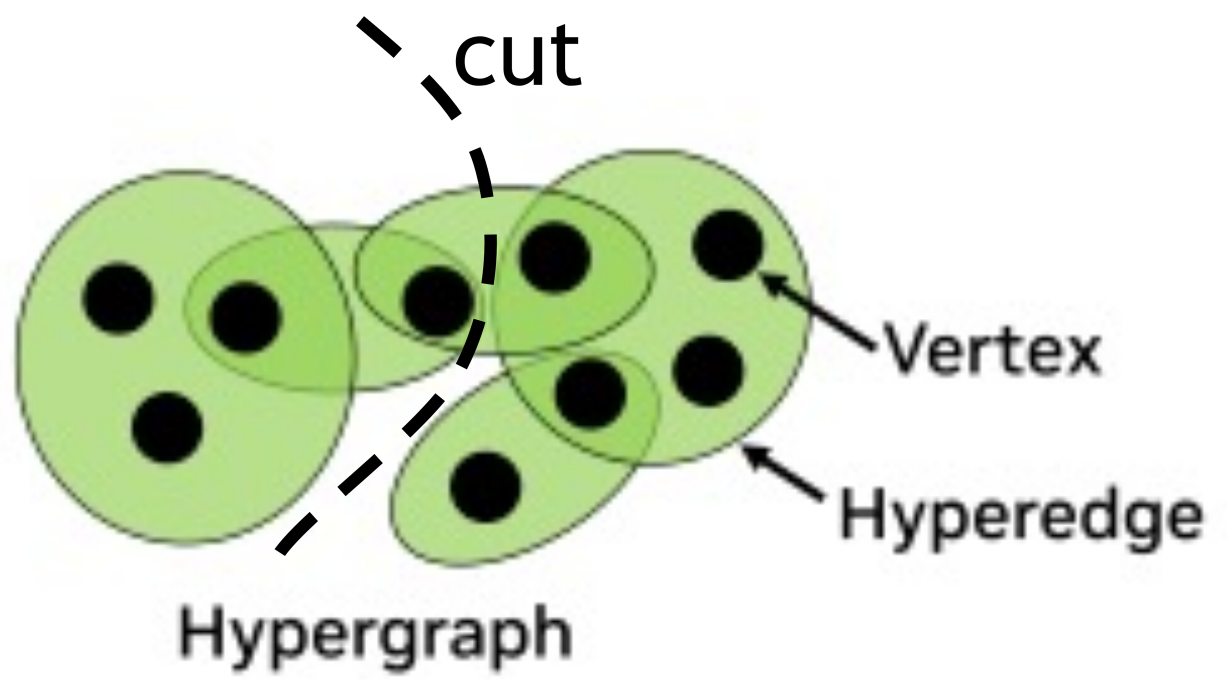
OUTLINE

- **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

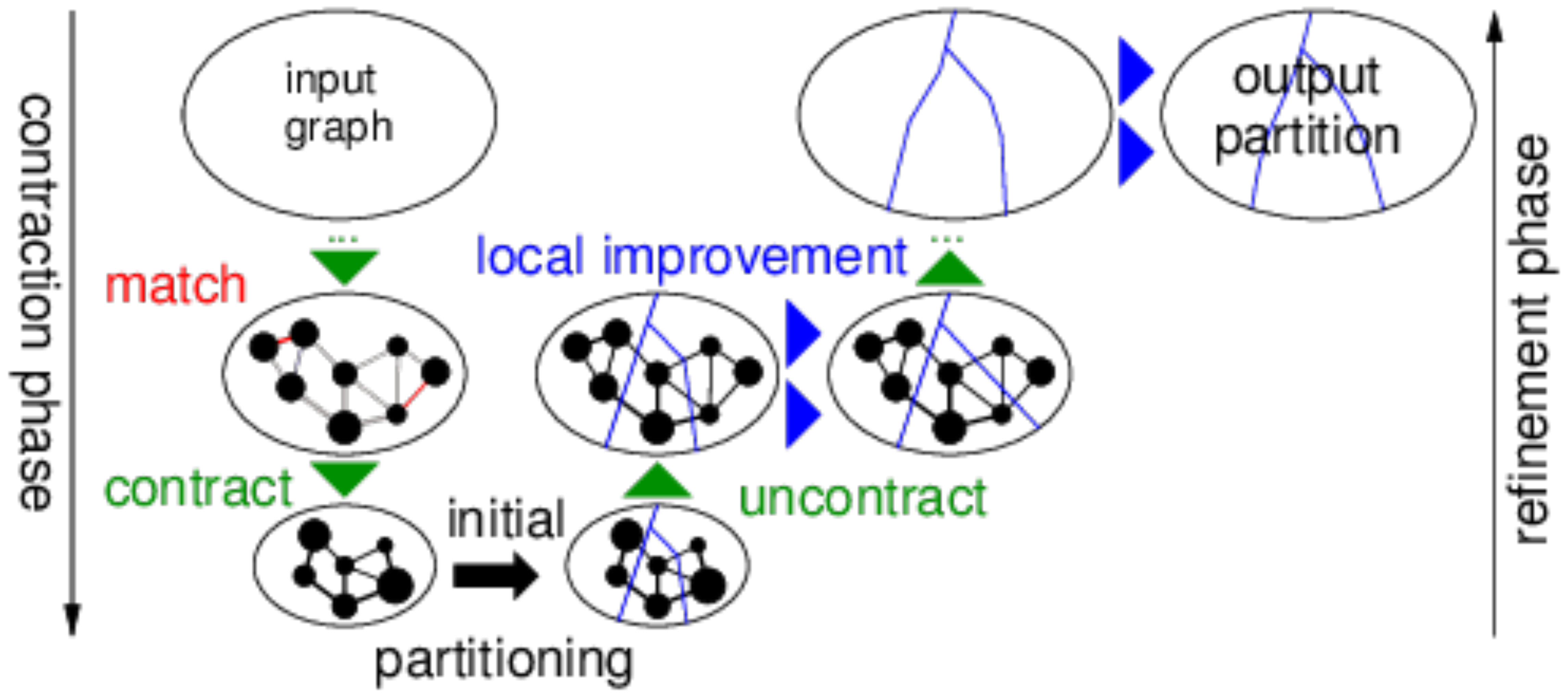
Motivations

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**



Hypergraph example



Multi-level graph partitioning

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 $\varepsilon \leq \frac{1}{k}$, letting $W = \sum_{v \in V} w_v$, the k-way balanced hypergraph partitioning problem can be mathematically formulated as:

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

Minimize cut size

$$\text{s.t.} \quad \cup_{i=1}^{k} V_i = V \quad \text{and} \quad V_i \cap V_j = \emptyset, \quad 0 \leq i, j \leq k \qquad (2)$$

Non-overlap between partition blocks

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

Balanced partition block sizes

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

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

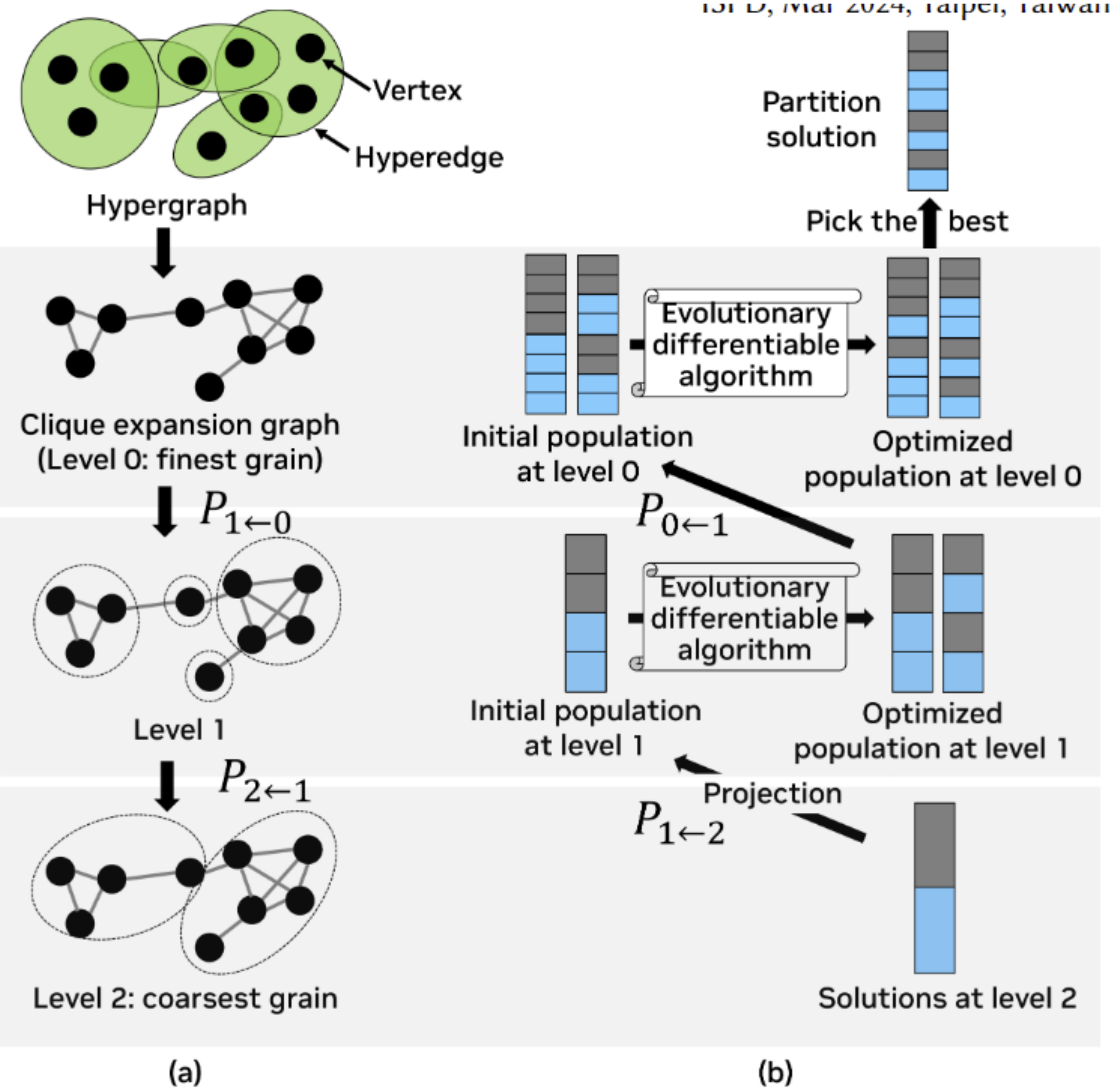


Figure 2: Overview of MedPart. (a) Spectral graph coarsening on a hypergraph. The hypergraph transformed to a clique expansion graph is progressively coarsened into several clusters for scalability. Projection matrices encoding the coarsening for use in (b) are built concurrently. (b) Multi-level optimization framework of MedPart. Partitioning assignments at coarser level l are used as a starting point for evolutionary differentiable optimization at finer level $l-1$.

[1] Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding, ICLR 2020

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

Algorithm 2: Evolutionary Differentiable Hypergraph Partitioning Algorithm

Input: I: number of generations; M: population size; Th: stagnation threshold
S: number of GD steps; T: checkpoint steps
$X_0 \leftarrow \{x_1^0, x_2^0, \dots, x_M^0\}$: initial population
Output: x^* : best partitioning solution
/* Evaluation */
$scores(X_0)$ = EvalFitness(X_0)
/* I generations */
for $i \leftarrow 1$ *to* I **do**
/* GA iteration */
/* Generate offspring population by crossover and mutation */
$\{c_1^i, c_2^i, \dots, c_M^i\} \leftarrow$ GenOffspring($X_{i-1}, scores(X_{k-1})$)
/* Evaluation */
$scores(\{c_1^i, c_2^i, \dots, c_M^i\})$ = EvalFitness($\{c_1^i, c_2^i, \dots, c_M^i\}$)
/* in parallel by batching */
for $m \leftarrow 1$ *to* M **do**
/* GD epoch */
Initialize the best solution for the current GD run: $c^{*i}_m \leftarrow c^i_m$,
$scores(c^{*i}_m) \leftarrow scores(c^i_m)$
Select hyper-parameters π^i_m
Initialize continuous solution: $\tilde{c}^i_m \leftarrow$ Relax(c^i_m)
for $s \leftarrow 1$ *to* S **do**
GD update of $\tilde{c}^i_m$ with π^i_m
if (s MOD T == 0) *or* (s == S) **then**
$c^i_m \leftarrow$ Discretize($\tilde{c}^i_m$)
$scores(c^i_m)$ = EvalFitness(c^i_m)
if $scores(c^i_m)$ *better than* $scores(c^{*i}_m)$ **then**
$c^{*i}_m \leftarrow c^i_m$
$scores(c^{*i}_m) \leftarrow scores(c^i_m)$
end
end
end
end
/* Gather best solutions from GD outcome */
$C_i^* \leftarrow \{c^{*i}_1, c^{*i}_2, \dots, c^{*i}_M\}$
$scores(C_i^*) \leftarrow \{scores(c^{*i}_1), scores(c^{*i}_2), \dots scores(c^{*i}_M)\}$
/* Update population with deterministic crowding */
X_i, $scores(X_k) \leftarrow$
UpdatePopulation(X_{i-1}, $scores(X_{k-1})$, C_i^*, $scores(C_i^*)$)
/* Early stop criterion */
if *the best fitness score does not improve for over Th generations* **then**
$x^* \leftarrow$ best solution from X_i
return x^*
end
end
return the best solution x^* among X_I

- 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

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

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.

	Statistics		$\varepsilon = 2\%$					$\varepsilon = 10\%$				
Benchmark	|V|	|E|	*SOTA*	*Spec*	$hMETIS_5$	*MedPart*	$MedPart_{h5}$	*SOTA*	*Spec*	$hMETIS_5$	*MedPart*	$MedPart_{h5}$
IBM01	12752	14111	200	202	213	202	205	166	171	190	166	166
IBM02	19601	19584	307	336	339	352	339	262	262	262	264	262
IBM03	23136	27401	951	959	972	955	957	950	952	960	955	954
IBM04	27507	31970	573	593	617	583	584	388	388	388	389	388
IBM05	29347	28446	1706	1720	1744	1748	1744	1645	1688	1733	1675	1668
IBM06	32498	34826	962	963	1037	1000	1012	728	733	760	788	760
IBM07	45926	48117	878	935	975	913	916	760	760	796	773	772
IBM08	51309	50513	1140	1146	1146	1158	1146	1120	1140	1145	1131	1135
IBM09	53395	60902	620	620	637	625	623	519	519	535	520	520
IBM10	69429	75196	1253	1318	1313	1327	1295	1244	1261	1284	1259	1257
IBM11	70558	81454	1051	1062	1114	1069	1067	763	764	782	774	765
IBM12	71076	77240	1919	1920	1982	1955	1949	1841	1842	1940	1914	1872
IBM13	84199	99666	831	848	871	850	850	655	693	721	697	696
IBM14	147605	152772	1842	1859	1967	1876	1884	1509	1768	1665	1639	1605
IBM15	161570	186608	2730	2741	2886	2896	2855	2135	2235	2262	2169	2166
IBM16	183484	190048	1827	1915	2095	1972	2095	1619	1619	1708	1645	1651
IBM17	185495	189581	2270	2354	2520	2336	2338	1989	1989	2300	2024	2028
IBM18	210613	201920	1521	1535	1587	1955	1587	1520	1537	1550	1829	1550
Avg gap to *SOTA*			0%	2.30%	6.20%	5.00%	3.70%	0%	2.10%	5.30%	3.40%	1.80%

Best published results

Leading partitioners

MedPart from scratch

MedPart refinement

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. $Spec_{h20}$ 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*.

	Statistics		$\varepsilon = 2\%$					$\varepsilon = 20\%$				
Benchmark	|V|	|E|	*SOTA*	$Spec_{h20}$	$hMETIS_5$	*MedPart*	$MedPart_{h5}$	*SOTA*	$Spec_{h20}$	$hMETIS_5$	*MedPart*	$MedPart_{h5}$
sparcT1_core	91976	92827	977	1012	1073	1067	1073	903	903	1290	624	624
neuron	92290	125305	239	252	276	262	271	206	206	270	271	270
stereo_vision	94050	127085	169	180	213	176	184	91	91	143	93	93
des90	111221	139557	372	402	372	390	372	358	358	441	349	357
SLAM_spheric	113115	142408	1061	1061	1061	1061	1061	1061	1061	1061	1061	1061
cholesky_mc	113250	144948	285	285	301	283	283	285	345	667	281	281
segmemtation	138295	179051	118	126	183	137	114	78	78	141	78	78
bitonic_mesh	192064	235328	584	587	667	594	595	483	483	590	511	493
dart	202354	223301	788	807	849	805	810	540	540	603	593	549
openCV	217453	284108	481	510	635	751	635	481	518	554	617	554
stap_qrd	240240	290123	398	399	399	386	386	295	295	295	297	287
minres	261359	320540	215	215	215	295	215	189	189	189	181	189
cholesky_bdti	266422	342688	1156	1156	1161	1172	1161	947	947	1024	1148	1024
denoise	275638	356848	416	416	916	695	516	224	224	478	228	224
sparcT2_core	300109	302663	1227	1244	1410	1329	1319	1227	1245	1972	1148	1081
gsm_switch	493260	507821	1827	1827	5974	1722	1714	1407	1407	5352	1503	1541
mes_noc	547544	577664	634	634	699	1320	699	617	617	633	1141	633
LU230	574372	669477	3273	3273	4070	3452	3480	2677	2677	3276	2720	2741
LU_Network	635456	726999	525	525	550	597	550	524	524	528	567	528
sparcT1_chip2	820886	821274	899	899	1524	1169	1129	783	783	1029	877	889
directrf	931275	1374742	574	574	646	771	646	295	295	379	317	337
bitcoin_miner	1089284	1448151	1297	1297	1570	1562	1570	1225	1225	1255	1282	1255
Avg gap to SOTA			0%	1.9%	31.0%	19.1%	7.6%	0%	1.4%	44.0%	8.3%	2.60%

Up to 30% smaller cut size

Runtime

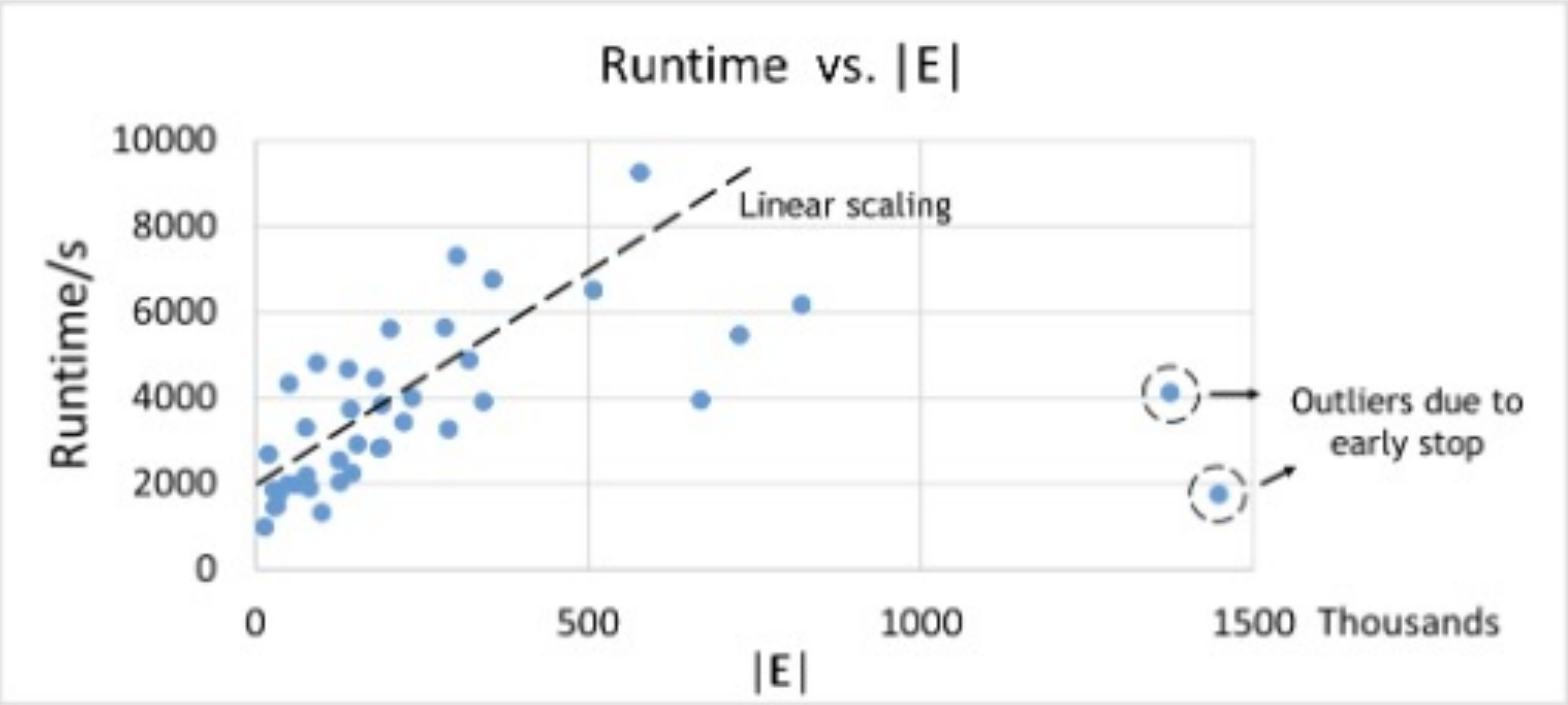


Figure 4: MedPart runtime on hypergraphs with different #edges.

Linear runtime scaling, but still large room to improve

Ablation Studies

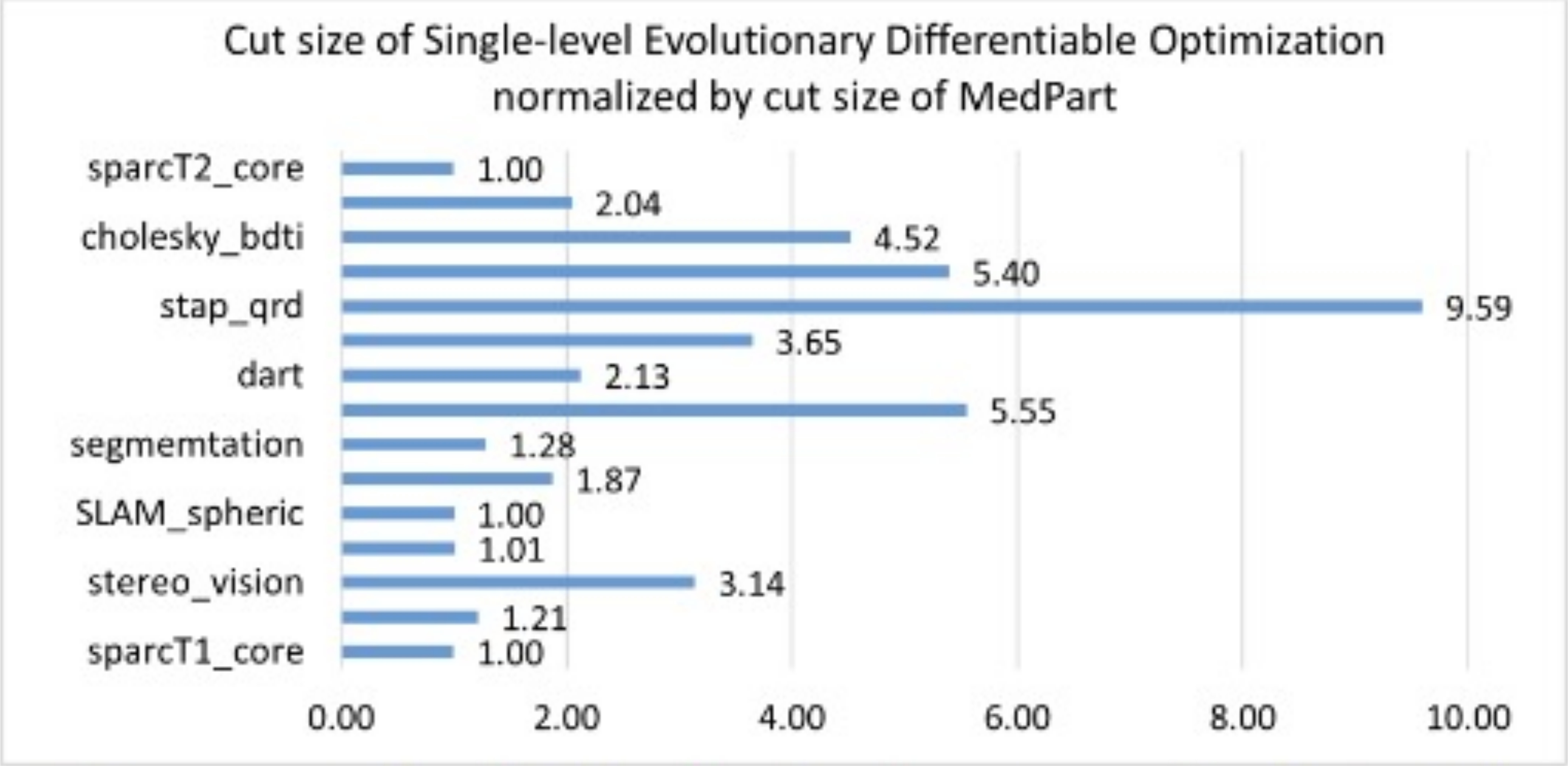
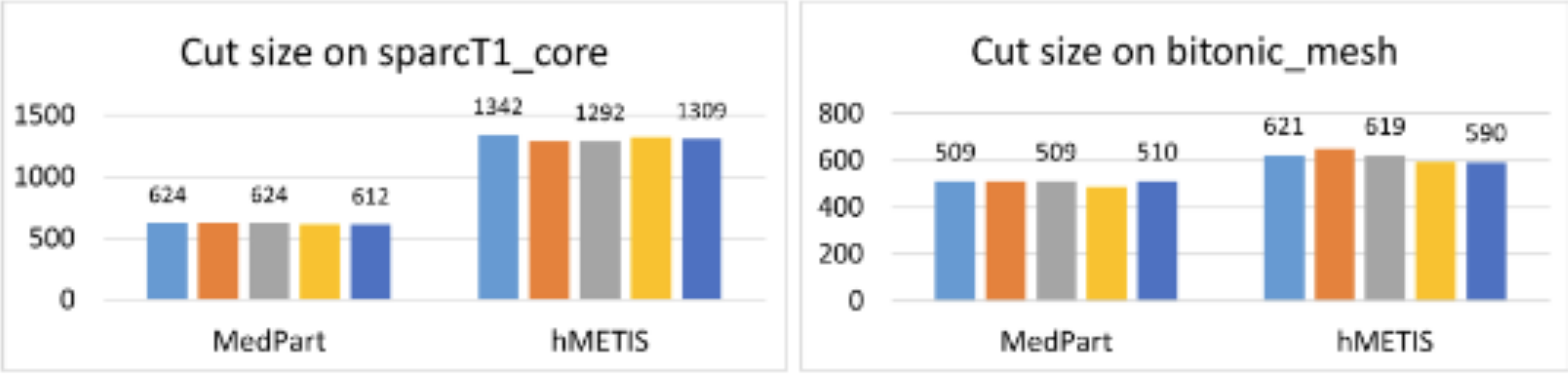


Figure 5: Impact of multi-level optimization on MedPart. The experiments are conducted on the top 15 benchmarks from the Titan23 benchmark suite, with ε set to 10%.

Multi-level paradigm is helpful



(a) (b)

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

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**

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 with100M vertices/edges
- Apply to other partitioning problem formulations, e.g., timing-driven netlist partitioning

NVIDIA

Looking forwards to your valuable feedback/comments!
(email: rliang@nvidia.com)