GPart: A GNN-Enabled Multilevel Graph Partitioner

Magi Chen and Ting-Chi Wang

National Tsing Hua University, Taiwan

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA sigda

Problem Formulation

Input

- A graph $G(V, E)$ which consists of a vertex set V and an edge set E.
- Each vertex v carries a weight w_v, and each edge e is associated with a weight w_e.
- A positive integer $k \geq 2$, and a balance factor $0 < \varepsilon \ll 1$.

We demonstrate the experimental results with uniform edge weights, uniform vertex weights, and for k=2 and k=3.

Objective

- Partition the vertex set V into a collection of k disjoint subsets $S = \{S_0, S_1, \dots, S_{k-1}\}$, where each S_i is referred to as a partition.
- The objective is to minimize the cut size, defined as:
  $cutsize(S) = \sum_{\{e=(u,v)\in E | u\in S_i, v\in S_j, and\ S_i \neq S_j\}} w_e$.

Constraint

Ensure that for each subset S_i:

$\sum_{v\in S_i} w_v \leq (1+\varepsilon)\frac{W}{k}$, for $0 \leq i \leq k-1$.

where $W = \sum_{v\in V} w_v$ is the weight of the entire vertex set.

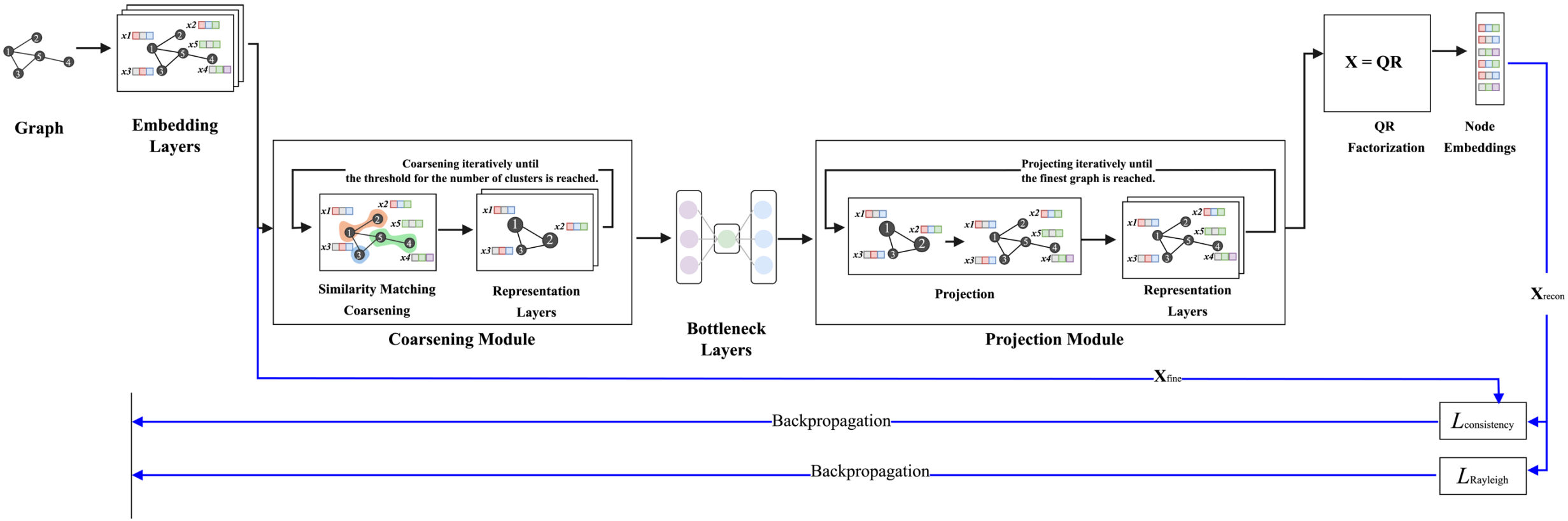
Overview of Existing Works

- **Multilevel:** Fast but limited in cut size quality (e.g., METIS [SIAM ’98], G-kway [DAC ’24])
  - G-kway: A GPU-accelerated multilevel graph partitioner that offers fast in execution but achieves cut sizes comparable to METIS.
- **GNN-based:** Good accuracy but poor scalability (e.g., GAP [ICLR ’19], GenPart [ICCAD ’24])
  - Requiring huge amounts of memory for handling large graphs.

Our Contributions

1. **GPart**: A multilevel graph partitioning framework using GNN-generated embeddings
2. **Centroid-guided refinement** to improve cut quality
3. **Scalable and memory-efficient**: 24.6x less memory than GAP
4. **Superior cut size** across Titan23 and DIMACS benchmarks

Framework Overview - Training



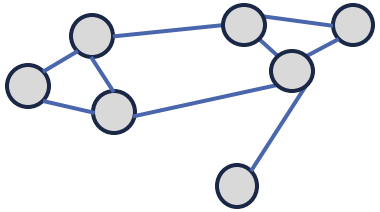
1. Training focuses on learning better embeddings
2. Once learned, embeddings guide the coarsening module
3. Bottleneck layers, projection module and QR factorization are "**Training-Time Only**"

Details of Components

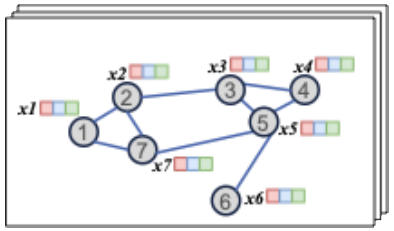
1. **Embedding Layers**
   - 3-layer GraphSAGE with 16-dim hidden layers
   - Input node feature: Standardized node index
   - Output: Embeddings with unit-length normalization
2. **Coarsening Module**
   - Employs GPU-accelerated heavy edge matching + sigmoid distance-based similarity
   - Reduces graph sizes to approximately 1/20 of the original
3. **Bottleneck and Projection**
   - **Bottleneck:** A sequence of linear layers
   - **Projection:** Unpooling from coarse to fine levels

Coarsening Module - Learning-Guided Coarsening

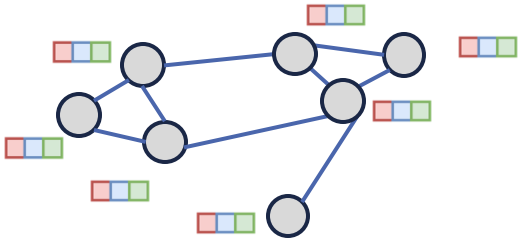
(1)Original graph
- Unit vertex weight
- Unit edge weight



(2)Embedding Layers



(3)Original graph + embeddings $\{x_u, x_v, ...\}$
- Unit vertex weight
- Unit edge weight



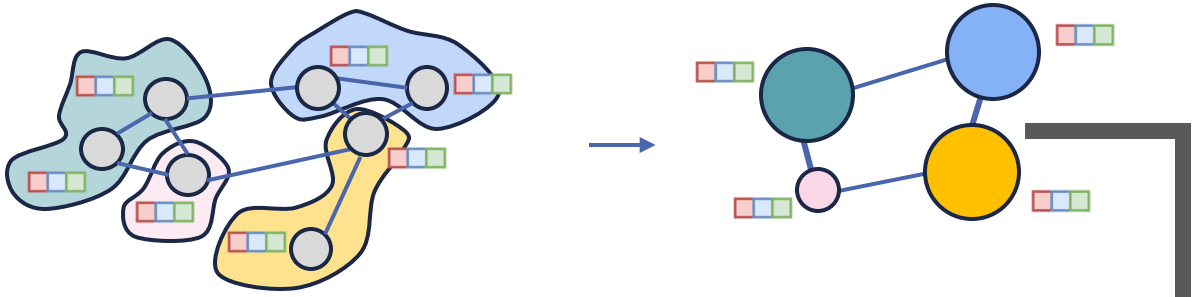
Super node's embedding is represented by mean of weighted embeddings.

(4) Update edge weights for coarsening guidance

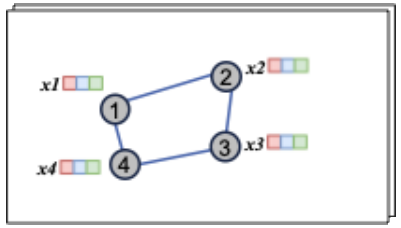
$$w_{uv} = sigmoid(-\|x_u - x_v\|_2) \times w_{uv}^{(orig)}$$

(*) Longer distance, lower similarity

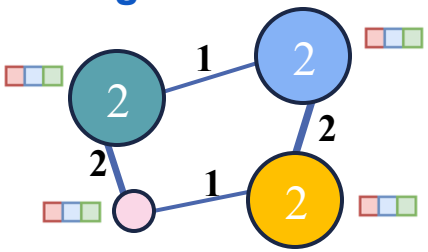
(5) GPU-Accelerated Greedy Matching



(6) Representation Layers



(7) Coarsened graph + embeddings
- varying vertex weights
- varying edge weights



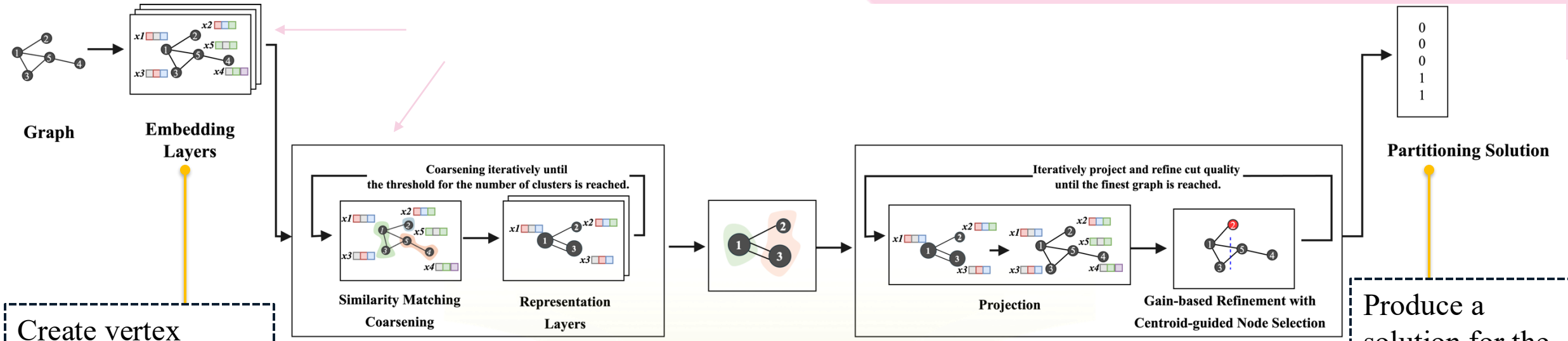
Repeat (4)~(7), until graph size is ~1/20 of the original

Loss Functions

1. $L_{consistency}$ **: Spectral Consistency Loss**
   - Matches Rayleigh quotients between embeddings before and after coarsening.
   - Preserves spectral properties throughout the coarsening and uncoarsening process.

2. $L_{Rayleigh}$**: Rayleigh Quotient Alignment Loss**
   - Aligns embeddings with the Fiedler vector
   - Captures higher-frequency spectral components (essential for multi-way partitioning)

Framework Overview – Evaluation



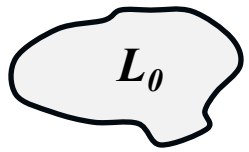
Create vertex embeddings for the original graph

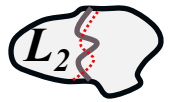
Iteratively coarsening using the coarsening module

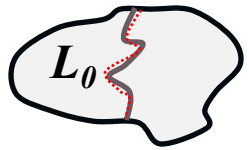
Use METIS on the coarsest graph

Iteratively projection and cut refinement

Produce a solution for the original graph







Mapped to a multilevel flow

Centroid-Guided, Gain-Based Refinement

Spectral (Embedding) Domain

① Utilize learned spectral embeddings to detect “candidate” nodes.
② A node is marked as a candidate if it is closer to another partition's centroid than its current one.

Graph (Structural) Domain

① **Gain Calculation:** Estimate the cut size reduction if a candidate node is moved.
② **FM-based Refinement:** Perform a one-pass FM operation **restricted to candidate nodes** with up to 50 moves per uncoarsening level.
③ **Best Partition Recorded:** Continuously record the configuration with the lowest cut size, and revert if necessary.

Memory Usage Comparison

	Statistics		Memory Usage (GB)		
Benchmark	$\|V\|$	$\|E\|$	*GAP*	*GenPart*	*GPart*
bitcoin_miner	1,089,284	3,662,788	20.61	10.51	0.81
bitonic_mesh	192,064	12,508,583	49.33	24.92	2.07
cholesky_bdti	266,422	4,660,080	19.54	9.89	0.79
cholesky_mc	113,250	2,592,938	10.67	5.39	0.44
dart	202,354	16,465,022	64.63	32.65	2.72
denoise	275,638	22,737,143	89.23	45.07	3.76
des90	111,221	6,752,436	26.68	13.48	1.12
directrf	931,275	19,432,048	80.41	40.68	3.26
gsm_switch	493,260	60,337,200	235.36	118.87	9.93
LU230	574,372	14,618,649	59.74	30.21	2.45
LU_Network	635,456	82,063,167	319.90	161.56	13.50
mes_noc	547,544	60,370,265	235.81	119.10	9.94
minres	261,359	6,495,763	26.58	13.44	1.09
neuron	92,290	2,770,749	11.23	5.68	0.47
openCV	217,453	28,852,387	112.44	56.78	4.75
segmentation	138,295	10,471,126	41.16	20.79	1.73
SLAM_spheric	113,115	5,636,907	22.39	11.31	0.94
sparcT1_core	91,976	16,697,490	64.87	32.76	2.75
sparcT2_core	300,109	46,519,712	180.99	91.40	7.65
stap_qrd	240,240	4,071,793	17.12	8.67	0.69
stereo_vision	94,050	3,785,776	15.15	7.65	0.64
Average Normalized Memory Usage			24.6x	12.4x	**1x**

GPart uses 24.6x less memory than GAP, 12.4x less than GenPart, as validated on Titan23 benchmarks.

Cut Size Results (Titan23)

GPart vs. METIS

	$k = 2,\ \varepsilon = 3\%$			$k = 2,\ \varepsilon = 10\%$		
Benchmark	*METIS*	*GPart*	**Improvement (%)**	*METIS*	*GPart*	**Improvement (%)**
bitcoin_miner	4,173	2,904	30.41%	4,169	2,628	36.96%
bitonic_mesh	142,095	150,766	-6.10%	142,427	140,738	1.19%
cholesky_bdti	37,439	19,066	49.07%	35,755	18,557	48.10%
cholesky_mc	14,882	4,435	70.20%	8,949	4,173	53.37%
dart	44,648	35,857	19.69%	47,938	36,453	23.96%
denoise	1,976	1,518	23.18%	1,785	1,151	35.52%
des90	142,863	77,042	46.07%	142,870	78,970	44.73%
directrf	30,901	8,042	73.97%	31,201	7,715	75.27%
gsm_switch	14,416	8,189	43.20%	20,062	6,370	68.25%
LU230	54,328	47,267	13.00%	54,047	42,828	20.76%
LU_Network	14,883	13,421	9.82%	12,495	7,911	36.69%
mes_noc	81,956	51,047	37.71%	81,966	36,143	55.90%
minres	10,915	7,951	27.16%	11,162	5,426	51.39%
neuron	6,349	2,926	53.91%	6,322	3,084	51.22%
openCV	28,222	20,190	28.46%	29,169	15,139	48.10%
segmentation	1,661	573	65.50%	7,799	501	93.58%
SLAM_spheric	5,291	5,164	2.40%	5,291	5,041	4.73%
sparcT1_core	19,039	15,509	18.54%	20,953	18,968	9.47%
sparcT2_core	27,217	26,976	0.89%	24,869	18,712	24.76%
stap_qrd	31,383	20,882	33.46%	26,679	17,831	33.16%
stereo_vision	1,212	288	76.24%	1,821	289	84.13%
Average Cut Size Improvement			**34.13%**			**42.92%**

	$k = 3,\ \varepsilon = 3\%$			$k = 3,\ \varepsilon = 10\%$		
Benchmark	*METIS*	*GPart*	**Improvement (%)**	*METIS*	*GPart*	**Improvement (%)**
bitcoin_miner	9,650	8,227	14.75%	10,350	8,250	20.29%
bitonic_mesh	230,579	212,809	7.71%	194,302	200,764	-3.33%
cholesky_bdti	53,503	31,405	41.30%	53,959	26,452	50.98%
cholesky_mc	21,754	7,835	63.98%	20,617	7,351	64.34%
dart	66,685	61,514	7.75%	70,156	50,369	28.20%
denoise	12,742	4,359	65.79%	9,878	4,253	56.94%
des90	204,411	106,937	47.69%	201,682	101,921	49.46%
directrf	38,632	23,860	38.24%	39,245	20,173	48.60%
gsm_switch	26,532	11,829	55.42%	28,218	12,118	57.06%
LU230	102,335	88,965	13.06%	95,167	83,638	12.11%
LU_Network	31,830	26,736	16.00%	29,255	19,858	32.12%
mes_noc	103,146	84,880	17.71%	111,964	83,234	25.66%
minres	18,736	12,392	33.86%	17,931	11,426	36.28%
neuron	8,597	5,352	37.75%	9,331	4,887	47.63%
openCV	133,203	58,412	56.15%	96,606	50,337	47.89%
segmentation	8,031	1,880	76.59%	3,798	1,858	51.08%
SLAM_spheric	31,396	32,577	-3.76%	34,734	30,610	11.87%
sparcT1_core	37,576	35,529	5.45%	35,797	26,465	26.07%
sparcT2_core	46,023	39,660	13.83%	45,776	40,966	10.51%
stap_qrd	42,891	32,511	24.20%	41,638	30,808	26.01%
stereo_vision	2,140	1,554	27.38%	2,499	894	64.23%
Average Cut Size Improvement			**31.47%**			**36.38%**

Cut Size Results (DIMACS)

GPart vs. G-kway on DIMACS

	Statistics		$k = 2,\ \varepsilon = 3\%$		
Benchmark	$\|V\|$	$\|E\|$	*G-kway* [37]	*GPart*	**Improvement (%)**
delaunay_n24	16,777,216	50,331,601	8,463	7,465	11.79%
ldoor	952,203	22,785,136	25,578	23,932	6.44%
NLR	4,163,763	12,487,976	4,705	3,812	18.98%
asia.osm	11,950,757	12,711,603	7	7	0.00%
Average Cut Size Improvement			-	-	**9.30%**

AI

Security

Systems

Thank You For Listening

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm