Adaptive Layout Decomposition with Graph Embedding Neural Networks

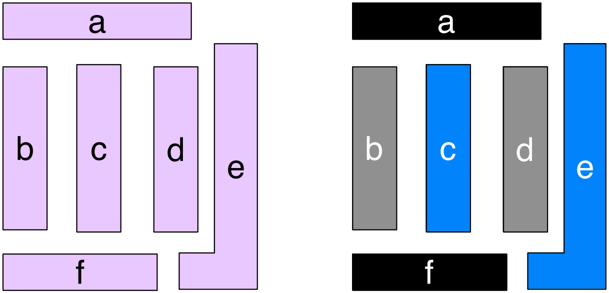
Wei Li[1], Jialu Xia[1], Yuzhe Ma[1], Jialu Li[1], Yibo Lin[2], Bei Yu[1]

[1]The Chinese University of Hong Kong
[2]Peking University

Outline

Outline

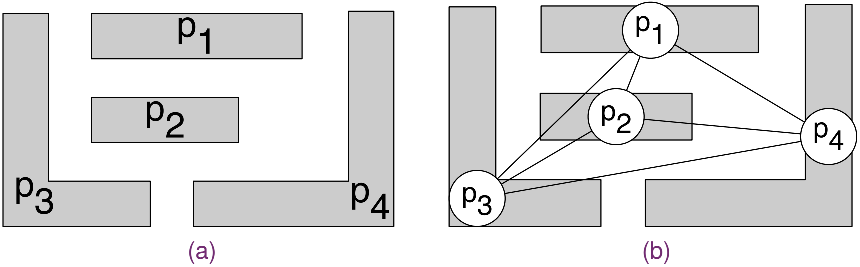
Multiple Patterning Lithography Decomposition



(a) (b)

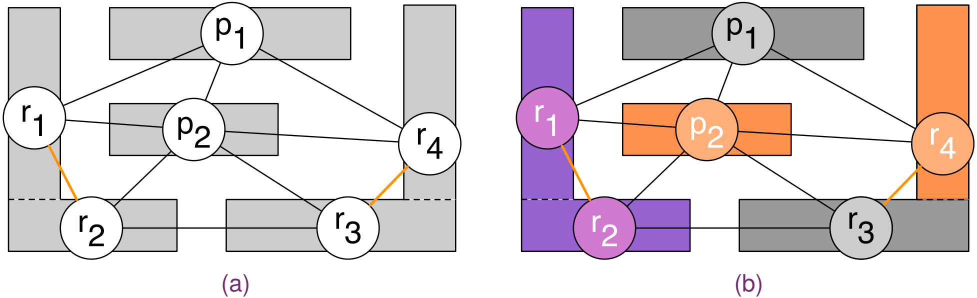
An example of the layout and corresponding decomposition results

Uncolorable case: Conflict



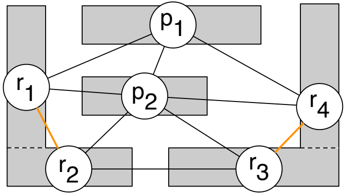
An example of the uncolorable case

One possible solution for the uncolorable case: Stitch



An example of the stitch candidate and stitch

Problem Formulation



$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & \sum c_{ij} + \alpha \sum s_{ij}, & & \text{(1a)}\\
\text{s.t.} \quad & c_{ij} = (x_i == x_j), & \forall e_{ij} \in CE, & \quad \text{(1b)}\\
& s_{ij} = (x_i \neq x_j), & \forall e_{ij} \in SE, & \quad \text{(1c)}\\
& x_i \in \{0, 1, \dots, k\}, & \forall x_i \in \boldsymbol{x}, & \quad \text{(1d)}
\end{aligned}
$$

$\boldsymbol{x}$: color assigned to each node, CE: conflict edge set, SE: stitch edge set.

Integer Linear Programming (ILP)∗

$$\min \sum_{e_{ij}\in \mathrm{CE}} c_{ij} + \alpha \sum_{e_{ij}\in \mathrm{SE}} s_{ij} \tag{2a}$$

$$\text{s.t. } x_{i1} + x_{i2} \le 1, x_{ij} \in \{0,1\}. \tag{2b}$$

$$x_{i1} + x_{j1} \le 1 + c_{ij1}, x_{i2} + x_{j2} \le 1 + c_{ij2}, \quad \forall e_{ij} \in \mathrm{CE}, \tag{2c}$$

$$(1 - x_{i1}) + (1 - x_{j1}) \le 1 + c_{ij1}, \quad \forall e_{ij} \in \mathrm{CE}, \tag{2d}$$

$$(1 - x_{i2}) + (1 - x_{j2}) \le 1 + c_{ij2}, \quad \forall e_{ij} \in \mathrm{CE}, \tag{2e}$$

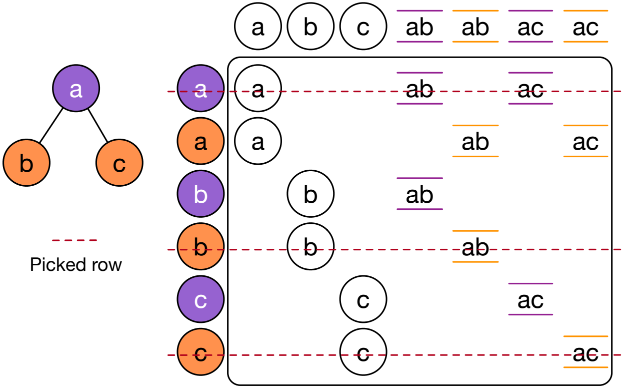
$$c_{ij1} + c_{ij2} \le 1 + c_{ij}, \quad \forall e_{ij} \in \mathrm{CE}, \tag{2f}$$

$$|x_{j1} - x_{i1}| \le s_{ij1}, |x_{j2} - x_{i2}| \le s_{ij2}, \quad \forall e_{ij} \in \mathrm{SE}, \tag{2g}$$

$$s_{ij} \ge s_{ij1}, s_{ij} \ge s_{ij2}, \quad \forall e_{ij} \in \mathrm{SE}, \tag{2h}$$

∗Bei Yu et al. (Mar. 2015). "Layout Decomposition for Triple Patterning Lithography". In: *IEEE TCAD* 34.3, pp. 433–446.

Exact Cover-based algorithm (EC)†



An example of the exact cover-based algorithm

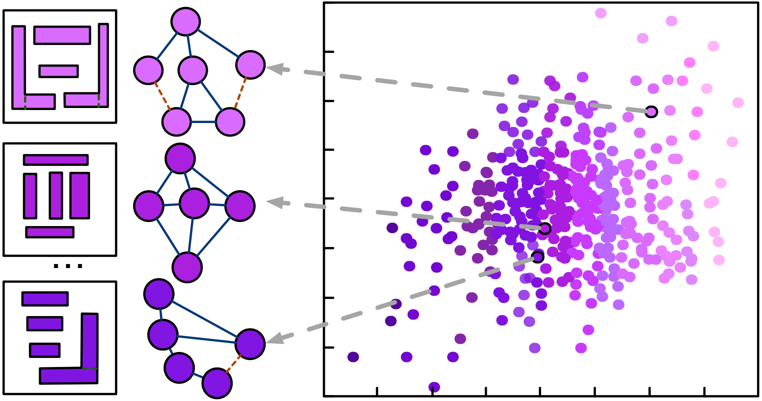
†Hua-Yu Chang and Iris Hui-Ru Jiang (2016). "Multiple patterning layout decomposition considering complex coloring rules". In: *Proc. DAC*, 40:1–40:6.

Pros and cons analysis

- ILP
  - Pros: Optimal
  - Cons: Bad runtime performance
- EC
  - Pros: High efficiency
  - Cons: Degradation of the solution quality
- Graph matching‡
  - Pros: Good performance in both efficiency and quality for small graphs
  - Cons: Graph library size is limited

‡Jian Kuang and Evangeline F. Y. Young (2013). "An Efficient Layout Decomposition Approach for Triple Patterning Lithography". In: *Proc. DAC*. San Francisco, California, 69:1–69:6.

Graph Embedding



An example of graph embeddings of layout graphs, where the graphs are transformed into vector space.

Graph Convolutional Network

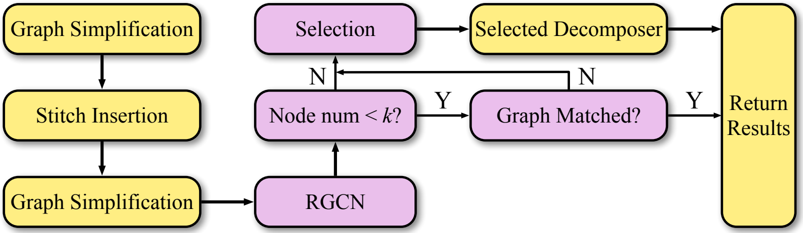
$$u_i^{(l+1)} = ReLU\left(\sum_{j \in N_i} W^{(l)} u^{(l)} + u_i^{(l)}\right), \tag{3}$$

$u^{(l)}$: node representation at the l_{th} layer, N_i: neighbours of node i.

- Composed of two modules, aggregator and encoder
- Node embedding: node representation at the final layer
- Graph embedding: obtained from node embedding through some operations such as summation and mean
- Not applicable for heterogeneous graphs

Outline

Framework Overview



The online workflow of our framework.

- Online: Shown in the figure.
- Offline: Model training & Graph library construction.

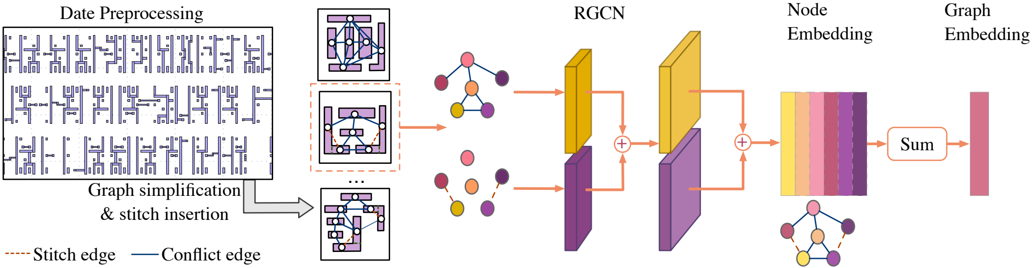
Relational Graph Convolutional Networks (RGCN)

$$\boldsymbol{u}_i^{(l+1)} = ReLU\left(\sum_{e \in E} \sum_{j \in N_i^e} \boldsymbol{W}_e^{(l)} \boldsymbol{u}_j^{(l)} + \boldsymbol{u}_i^{(l)}\right), \tag{4}$$

E:{CE (conflict edge set), SE (stitch edge set)}

- Neighbours connected by different kinds of edges are assigned to different encoder tracks.
- Applicable for heterogeneous graphs

Graph Embedding Workflow by RGCN



Overview of the process for graph embedding

Offline: Graph Library Construction

What we need?

- Enumerate all possible graphs under a size constraint
- Avoid isomorphic graphs

Rough Algorithm

1. Enumerate all valid graphs under the given size constraint
2. For each graph enumerated, calculate the graph embedding and normalize it
3. Multiply it with the graph embeddings in the library
4. If the maximal value is less than one, insert the graph and corresponding optimal solution by ILP into the library

Online: Graph Matching & Decomposer Selection

Graph Matching

- Similar idea with graph library construction
- Return the optimal solution of the corresponding matched graph whose graph embedding multiplication result is exactly one

Decomposer Selection

$$y = \arg\max(\boldsymbol{W}_s \boldsymbol{h} + \boldsymbol{b}_s), \tag{5}$$

W_s, b_s: trainable weight and bias, $\boldsymbol{h}$: graph embedding
Two-class classification problem: ILP or EC

Outline

Effectiveness of RGCN

		Label ILP	Label EC
Predicted	ILP	13	682
	EC	0	5900
Recall		100.0%	
F1-score		0.0367	

(a) Proposed RGCN

		Label ILP	Label EC
Predicted	ILP	2	244
	EC	11	6338
Recall		15.4%	
F1-score		0.0154	

(b) Conventional GCN

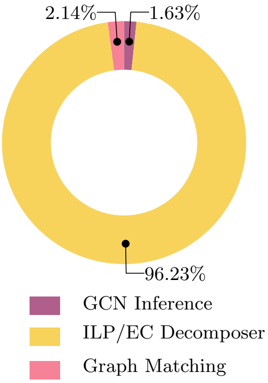
- Classify all ‘ILP’ cases correctly and such achieves the optimality
- $2\times$ F1-score, $6\times$ Recall

Comparison with state-of-the-art

Circuit	ILP				SDP				EC				RGCN			
	st#	cn#	cost	time (s)	st#	cn#	cost	time (s)	st#	cn#	cost	time (s)	st#	cn#	cost	time (s)
C432	4	0	0.4	0.486	4	0	0.4	0.016	4	0	0.4	0.005	4	0	0.4	0.007
C499	0	0	0	0.063	0	0	0	0.018	0	0	0	0.011	0	0	0	0.015
C880	7	0	0.7	0.135	7	0	0.7	0.021	7	0	0.7	0.010	7	0	0.7	0.014
C1355	3	0	0.3	0.121	3	0	0.3	0.024	3	0	0.3	0.011	3	0	0.3	0.015
C1908	1	0	0.1	0.129	1	0	0.1	0.024	1	0	0.1	0.017	1	0	0.1	0.031
C2670	6	0	0.6	0.158	6	0	0.6	0.044	6	0	0.6	0.035	6	0	0.6	0.046
C3540	8	1	1.8	0.248	8	1	1.8	0.086	8	1	1.8	0.032	8	1	1.8	0.038
C5315	9	0	0.9	0.226	9	0	0.9	0.106	9	0	0.9	0.039	9	0	0.9	0.049
C6288	205	1	21.5	5.569	203	4	24.3	0.648	203	5	25.3	0.151	205	1	21.5	0.154
C7552	21	1	3.1	0.872	21	1	3.1	0.157	21	1	3.1	0.071	21	1	3.1	0.111
S1488	2	0	0.2	0.147	2	0	0.2	0.031	2	0	0.2	0.013	2	0	0.2	0.016
S38417	54	19	24.4	7.883	48	25	29.8	1.686	54	19	24.4	0.329	54	19	24.4	0.729
S35932	40	44	48	13.692	24	60	62.4	5.130	46	44	48.6	0.868	40	44	48	1.856
S38584	117	36	47.7	13.494	108	46	56.8	4.804	116	37	48.6	0.923	117	36	47.7	1.840
S15850	97	34	43.7	11.380	85	46	54.5	4.320	100	34	44	0.864	97	34	43.7	1.792
average			12.893	3.640			15.727	1.141			13.267	0.225			12.893	0.448
ratio			1.000	1.000			1.220	0.313			1.029	0.062			1.000	0.123

- Obtain the optimal solution in the benchmark
- Runtime is reduced to 12.3% compared to another optimal ILP-based algorithm

Runtime breakdown of our framework



- The decomposition runtime by the selected decomposer is the major bottleneck
- RGCN inference and graph matching runtime of our framework are actually trivial
- Our method has strong scalability and can be applied to select other more efficient decomposers in the future.

Outline

Conclusion

- Graph embedding by RGCN
  - Build the isomorphism-free graph library
  - Match graphs in the library
  - Adaptively select decomposer
- The results show that:
  - The obtained graph embeddings have powerful representation capability
  - Excellent balance between decomposition quality and efficiency
  - Our framework has strong scalability for future incremental selection

Thank You

Wei Li (wli@cse.cuhk.edu.hk)

Yuzhe Ma (yzma@cse.cuhk.edu.hk)

Yibo Lin (yibolin@pku.edu.cn)

Bei Yu (byu@cse.cuhk.edu.hk)