A Reinforcement Learning Agent for Obstacle-Avoiding Rectilinear Steiner Tree Construction

Po-Yan Chen[1], Bing-Ting Ke[2], Tai-Cheng Lee[2], I-Ching Tsai[3] , Tai-Wei Kung[3] , Li-Yi Lin[3] , En-Cheng Liu[3] , Yun-Chih Chang[3] , Yih-Lang Li[2], and Mango C.-T. Chao[1]

[1] Institute of Electronics,
[2] Institute of Computer Science and Engineering,
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

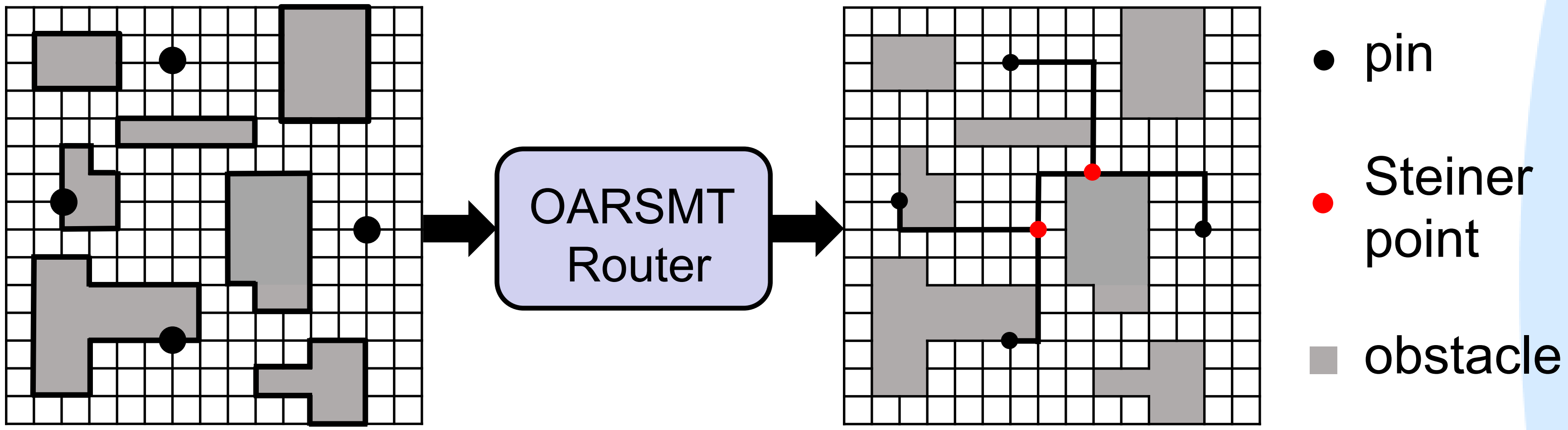
[3] Realtek Semiconductor Corporation
Hsinchu, Taiwan

Outline

- Problem Formulation and Objective
- Proposed OARSMT Router
- Concurrent Agent for OARSMT Router
- Data Augmentation for Training Time Reduction
- Conclusion

Obstacle-Avoiding Rectilinear Steiner Minimum Tree (OARSMT)

- OARSMT is a classic algorithmic problem that targets on finding a minimum-length routing tree that can connect all given pins and selected Steiner points with vertical and horizontal lines while not crossing given obstacles



- The Steiner tree problem is already known as NP-complete, and its advanced version, OARSMT, is considered even more difficult

Objective

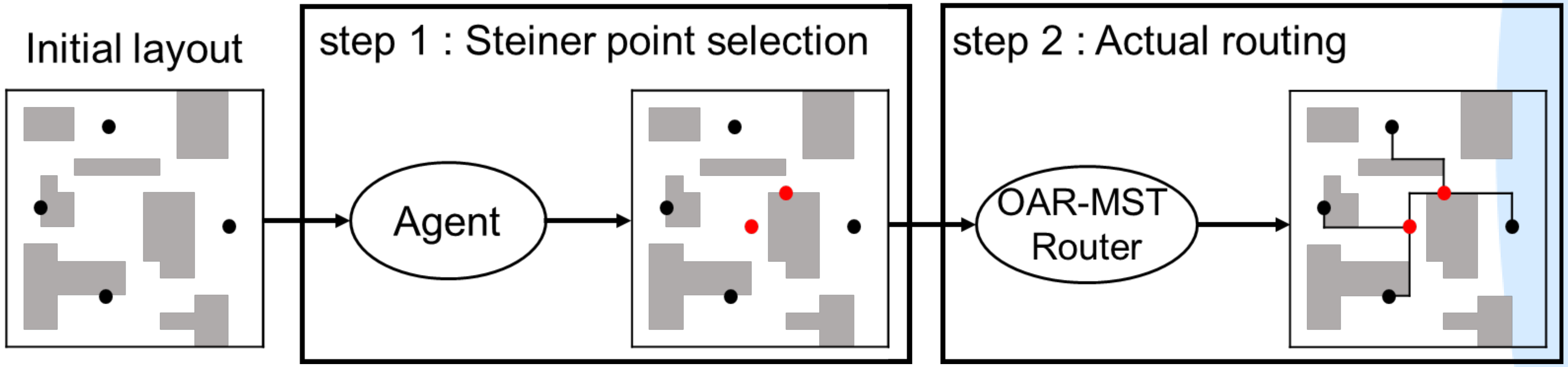
- Instead of manually developing another OARSMT algorithm, our objective is to propose a reinforcement-learning (RL) framework in which an OARSMT algorithm represented by an agent can be automatically developed and continually improved by itself.

Outline

- Problem Formulation and Objective
- Proposed OARSMT Router
- Concurrent Agent for OARSMT Router
- Data Augmentation for Training Time Reduction
- Conclusion

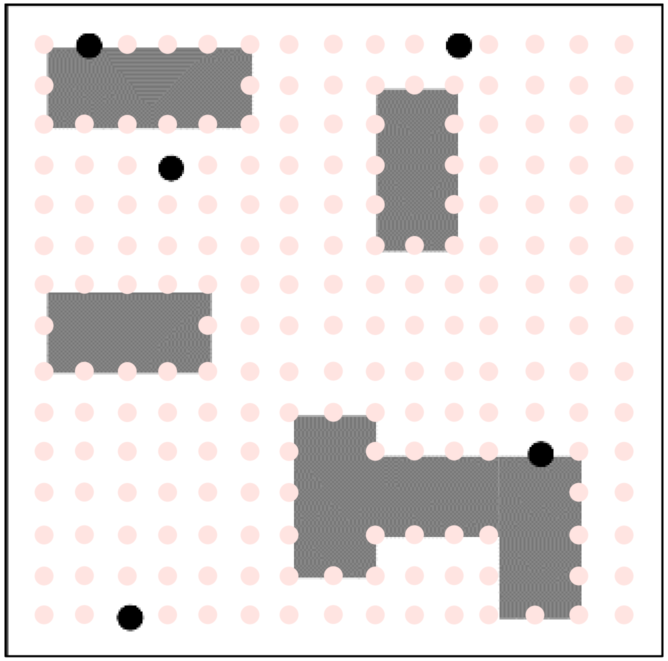
RL OARSMT Router

- Our OARSMT Router can be divided into two-step process
  - Select an optimal set of Steiner points based on a given layout
  - Build an obstacle avoiding rectilinear minimum spanning tree (OAR-MST) connecting all the pins and selected Steiner points, which can be done in polynomial time



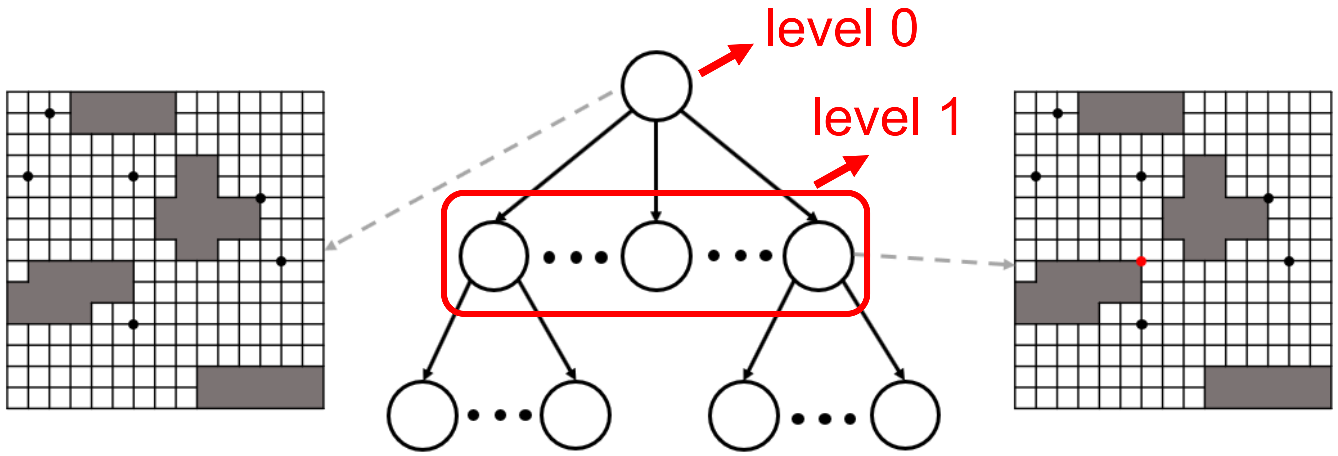
Sequential Agent

- Iteratively select the best next Steiner point on the current layout
- Input：a layout, represented by a HxVx3 integer array
- Output：a policy
  - probability of each vertex being the next selected Steiner point

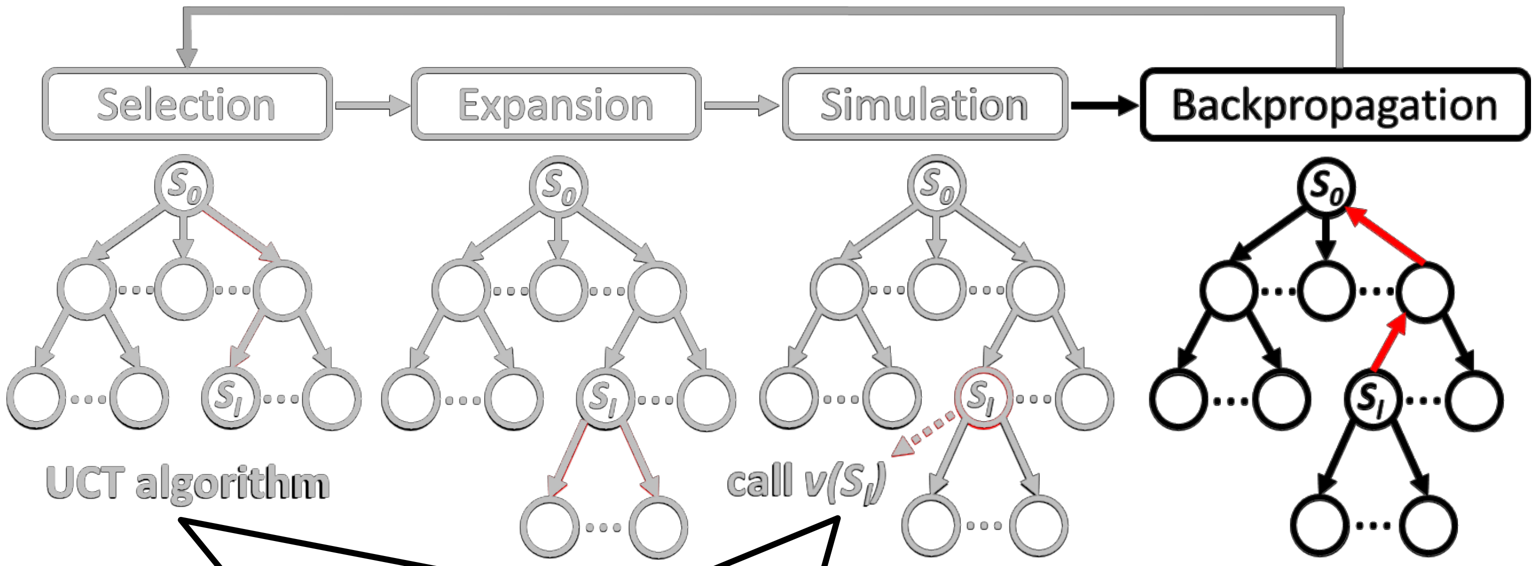


Monte Carlo Tree Search (MCTS)

- Our agent is built in the form of a neural network and trained by MCTS, which generates training sample with a more advanced policy as label by actually placing and evaluating various Steiner points on various layouts based on the current policy.
- In a Monte Carlo (MC) search tree
  - node：state or layout
  - edge：an action to insert a new Steiner point to the state



Exploration in MCTS



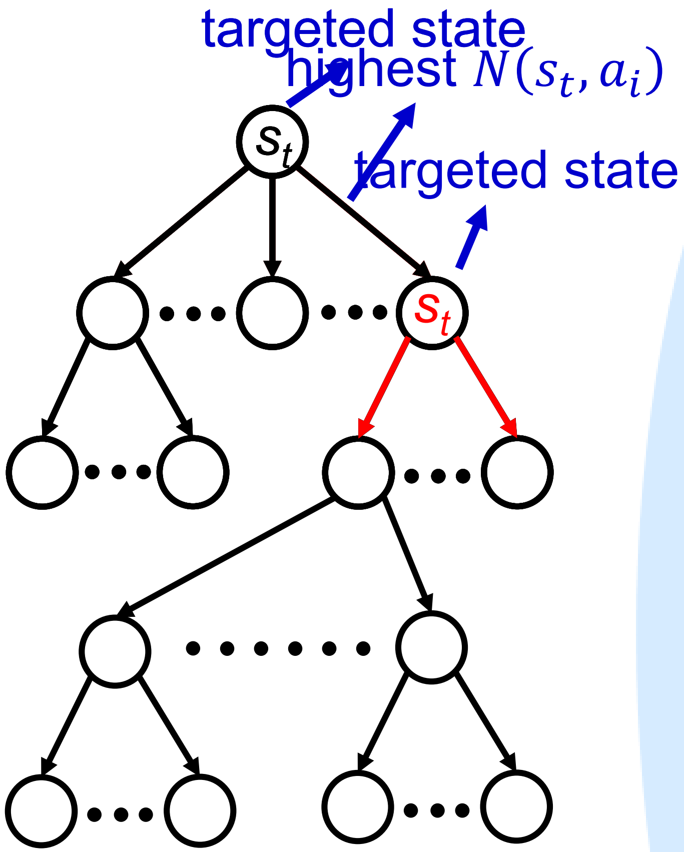
- a_i = ar
- $U(s_j, a_i) = P(s_j, a_i) * (\sum_i N(s_j, a_i))^{1/2} / (1 + N(s_j, a_i))$

- $v(s_l) = (wl_{s_0} - wl_{s_l}) / wl_{s_0}$

- Each edge (s_j, a_i) stores four records
  - $N(s_j, a_i)$: number of explorations passing the edge
  - $P(s_j, a_i)$: prior probability of the edge
  - $W(s_j, a_i)$: sum of the values of all traversed downstream edges
  - $Q(s_j, a_i)$: average of the values of the traversed downstream edges

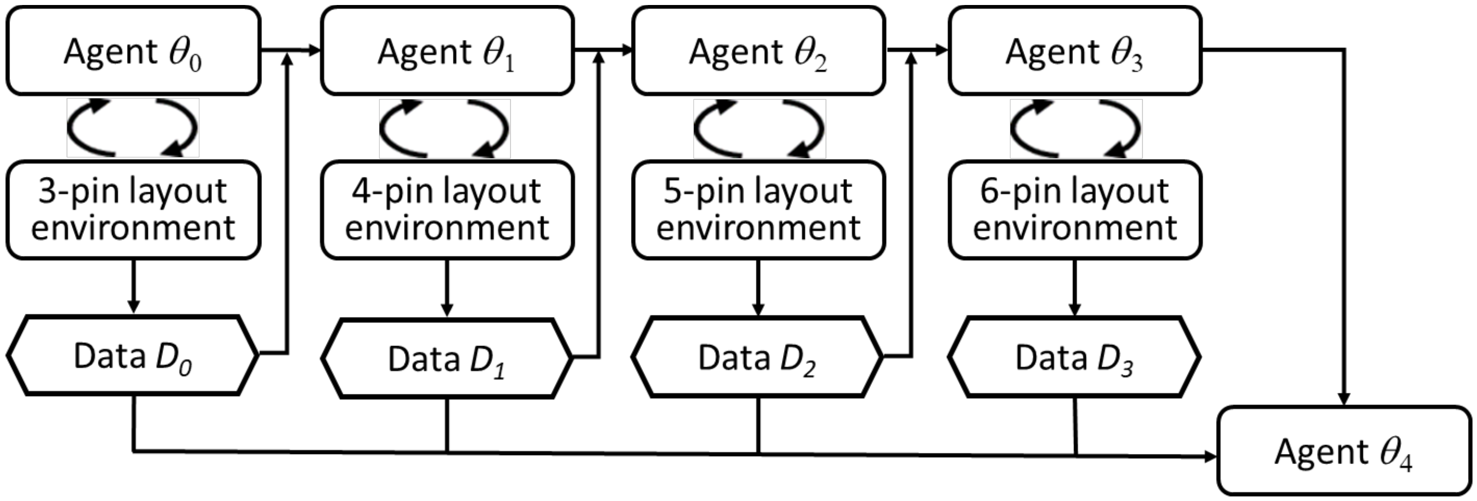
After α Explorations of MCTS

- A training sample can be created
  - Input：The layout which is stored in the targeted state
  - Label：A policy with probabilities which are determined by the frequency of all the edges along the targeted state
- MCTS will select a new targeted state (highest $N(s_t, a_i)$) and repeat explorations starting from the new targeted state



Techniques for Building Our Training Schedule

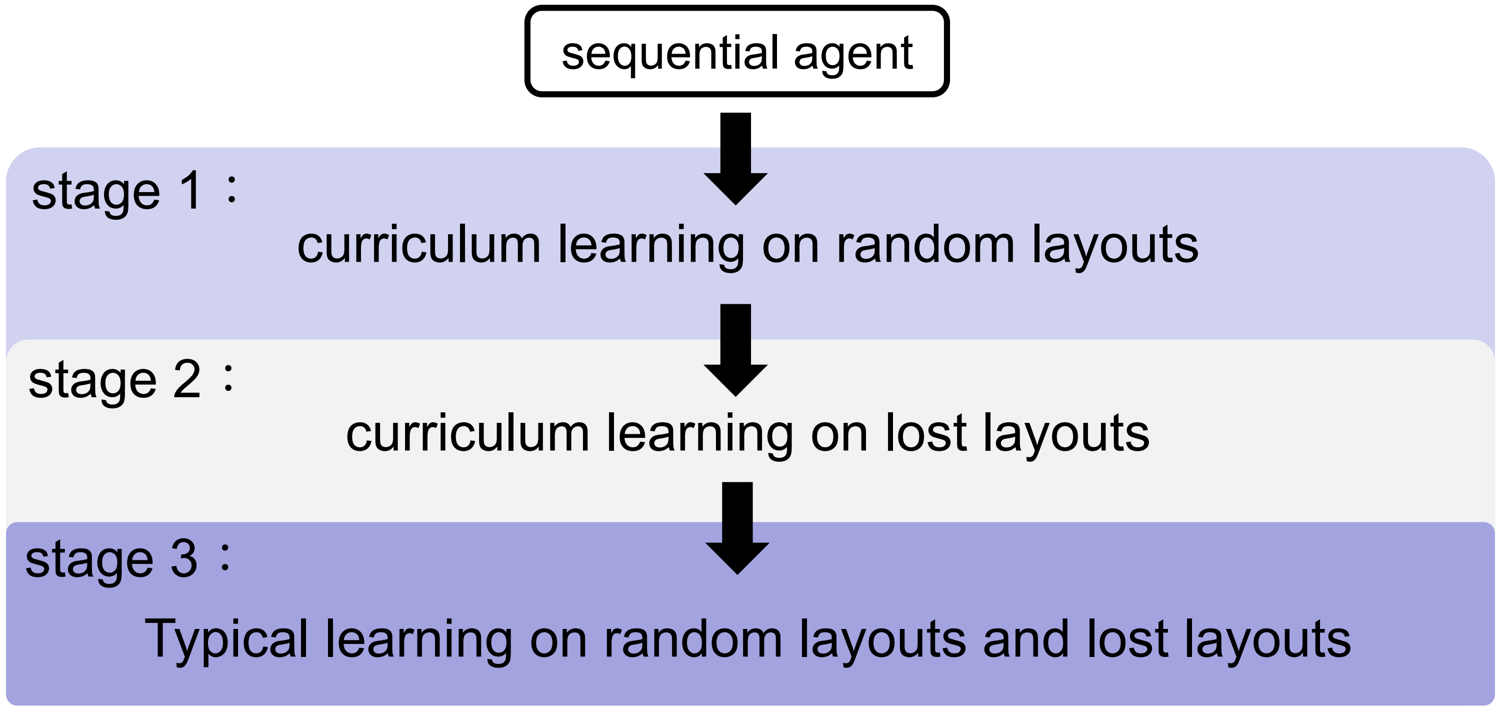
- We apply the concept of curriculum learning and learning from opponents for designing the training schedule
  - Curriculum learning：Plan the training of a RL agent with a curriculum that starts from easy cases to difficult ones



  - Learning from opponent：Identify the layouts on which the agent is currently lost to a given opponent and then ask MCTS to generate training samples based on those lost layouts

Our Training Schedule

- Training process of our reinforcement learning will be divided into three stages, and the sequential agent will be trained sequentially in the order of the three stages


sequential agent
stage 1：
curriculum learning on random layouts
stage 2：
curriculum learning on lost layouts
stage 3：
Typical learning on random layouts and lost layouts

Result of Sequential Agent vs. [8]

- Details of testing data
  - # of each pins of data：100,000
  - # of obstacles in 15x15 and 30x30 testing data are 8 and 12 respectively

layout dimensions	# of pins	total wire-length		avg. imp. (a-b)/a	our wire length vs. [8]'s	
		[8] (a)	Ours (b)		win rate	lost rate
15x15	3	1,637,778	1,622,592	0.927%	8.90%	0.26%
	4	2,107,787	2,082,896	1.181%	15.75%	0.61%
	5	2,506,130	2,469,119	1.477%	24.31%	1.26%
	6	2,852,402	2,809,524	1.503%	29.33%	2.17%
	Avg.	2,276,024	2,246,033	1.318%	19.57%	1.08%

layout dimensions	# of pins	total wire-length		avg. imp. (a-b)/a	our wire length vs. [8]'s	
		[8] (a)	Ours (b)		win rate	lost rate
30x30	3	3,212,731	3,177,044	1.111%	13.01%	1.33%
	4	4,122,114	4,064,686	1.393%	22.46%	2.33%
	5	4,874,367	4,793,290	1.663%	32.57%	3.62%
	6	5,536,808	5,441,344	1.724%	39.18%	4.95%
	Avg.	4,436,505	4,369,091	1.520%	26.80%	3.06%

[8] K.-W. Lin *et al.*, "A maze routing-based methodology with bounded exploration and path-assessed retracing for constrained multilayer obstacle-avoiding rectilinear Steiner tree construction," *ACM TODAES*, May, 2018

Result of More Pins than Trained

- When the number of pins exceeds that used in the training data, our sequential agents can still outperform [8] by a significant margin for every 15x15 or 30x30 cases.

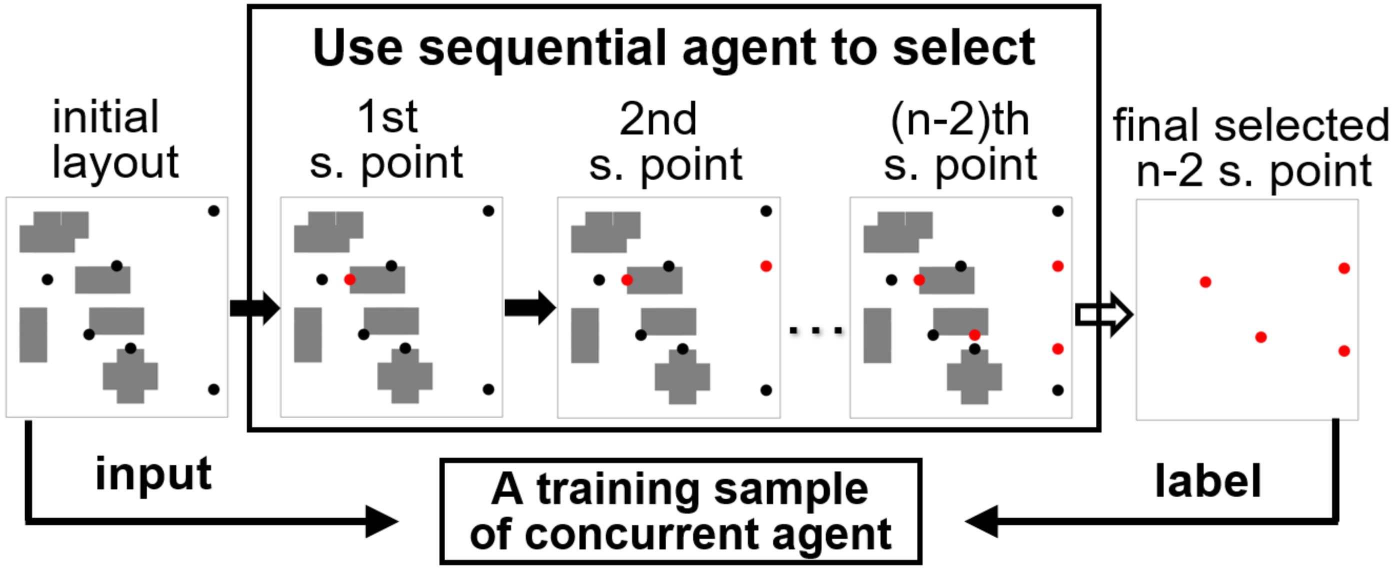
layout dimensions	# of obstacles	# of pins	avg. imp. over [8]	our wire length vs. [8]'s	
				win rate	lost rate
15x15	8	7	1.480%	33.09%	3.36%
		8	1.421%	35.89%	4.63%
		9	1.328%	37.62%	6.07%
		10	1.241%	39.43%	7.70%
		11	1.127%	40.15%	9.42%
		12	1.050%	41.22%	10.97%
30x30	12	7	1.685%	43.89%	6.53%
		8	1.630%	47.52%	7.92%
		9	1.571%	50.10%	9.18%
		10	1.488%	51.90%	10.71%
		11	1.415%	53.46%	11.83%
		12	1.348%	54.51%	12.89%

Outline

- Problem Formulation and Objective
- Proposed OARSMT Router
- Concurrent Agent for OARSMT Router
- Data Augmentation for Training Time Reduction
- Conclusion

Training a Concurrent Agent

- Training a concurrent agent is supervised learning while the label of each training sample is generated by the sequential agent
- All Steiner points selected by the sequential agent form the label of the training sample associated with the initial layout, where a vertex with and without a Steiner point is labeled as 1/(*n*-2) and 0


Use sequential agent to select
initial layout
1st s. point
2nd s. point
(n-2)th s. point
final selected n-2 s. point
input
A training sample of concurrent agent
label

Result of Concurrent Agent vs. [8]

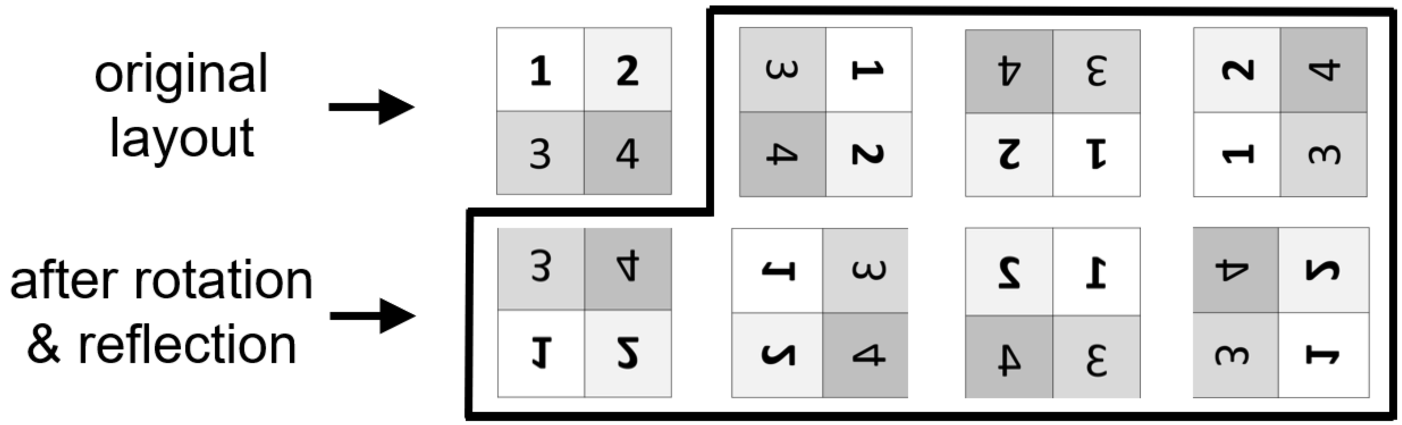
layout dimensions	# of pins	avg. imp. over [8]	average runtime per 100k layouts		
			ours (a)	[8] (b)	b / a
15x15	3~6	1.302%	22.90 s	30.22 s	1.32
30x30		1.485%	38.87 s	51.52 s	1.33

layout dimensions	# of pins	avg. imp. over [8]	our runtime		[8]'s runtime (b)	speed up b / a
			s. point selection	total (a)		
30x30	3	1.110%	28.42 s	34.80 s	30.34 s	0.87
	4	1.370%	28.50 s	37.61 s	44.44 s	1.18
	5	1.626%	28.59 s	40.14 s	58.53 s	1.46
	6	1.669%	28.70 s	42.91 s	72.77 s	1.70
	7	1.639%	28.90 s	44.04 s	85.63 s	1.94
	8	1.584%	29.03 s	45.99 s	100.07 s	2.18
	9	1.531%	29.11 s	48.00 s	113.50 s	2.36
	10	1.463%	29.31 s	51.16 s	128.22 s	2.51
	11	1.404%	29.33 s	52.45 s	149.84 s	2.86
	12	1.346%	29.54 s	54.46 s	165.79 s	3.04

Outline

Symmetric Data Augmentation

- The concept is to create and include geometrically symmetric training samples by rotating and/or reflecting both the input layout and the output label (policy) of a training sample generated by MCTS
  - A rotation can be 90°, 180° or 270°
  - A reflection can be x-axis based or y-axis based



Impact of Data Augmentation

- The total training time for such a 15x15 and 30x30 agents with using data augmentation (using exactly the same number of samples as for training the original ones) are 12.9% and 15.5% of that for the original ones
- The 15x15 and 30x30 agents with using data augmentation can achieve the same performance as the original agents in 59.2 and 275.1 hours

dimension	method	# of samples	avg. imp. over [8]	training time
15x15	w/o data aug.	1.18M	1.318%	269.4 hr.
	with data aug.	1.18M	1.290%	34.8 hr.
		1.88M	1.322%	59.2 hr.
30x30	w/o data aug.	1.2M	1.520%	1478.3 hr.
	with data aug.	1.2M	1.499%	228.8 hr.
		1.5M	1.524%	275.1 hr.

Conclusion

- Propose a policy-based RL framework used MCTS plus the UCT formula and applied curriculum learning and learning from opponents to train a policy agent that can tackle a classic algorithm problem in EDA, OARSMT
- A concurrent agent was trained to directly produce the end result of the sequential agent with only one model inference
- Data augmentation with symmetric samples was applied to further reduce the overall training time
- The experimental results based on 15x15 and 30x30 layouts have shown that our trained agent can outperform a state-of-the-art OARSMT [8] on both average wire length and runtime