NN-Steiner: A Neural-Algorithmic Approach for the Rectilinear Steiner Minimum Tree Problem

Andrew B. Kahng, Robert R. Nerem, Yusu Wang, Chien-Yi Yang
University of California San Diego

UC San Diego

TILOS

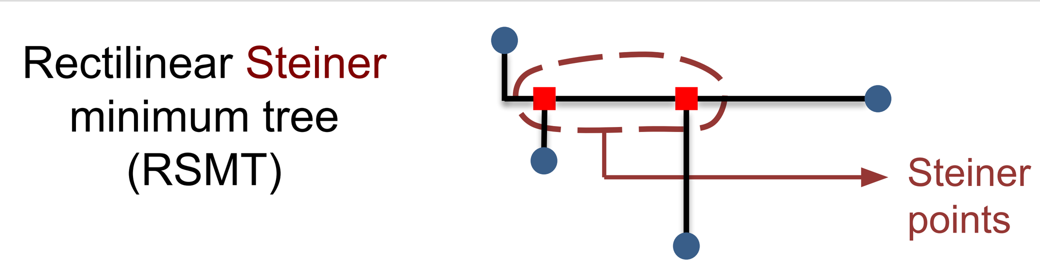
Overview

Main Idea: Combine algorithmic insights with neural networks

- **Motivation:** The rectilinear Steiner minimum tree (RSMT) problem, which is NP-hard, is fundamental to IC layout design
- Arora's algorithm for RSMTs achieves state-of-the-art (SOTA) theoretical guarantees, too costly for practice
- Our approach: NN-Steiner
  - Implementation of Arora's celebrated polynomial-time approximation scheme (PTAS) algorithm via a mixed-algorithmic-NN approach
  - **Replaces costly sub-algorithmic components with learning**, while keeping the DP framework
- NN-Steiner advantages:
  - Practical while still leveraging algorithmic insights
  - Uses bounded-size neural networks, thus efficient and effective to train
  - Learned sub-algorithmic components generalize to larger point sets than seen in training

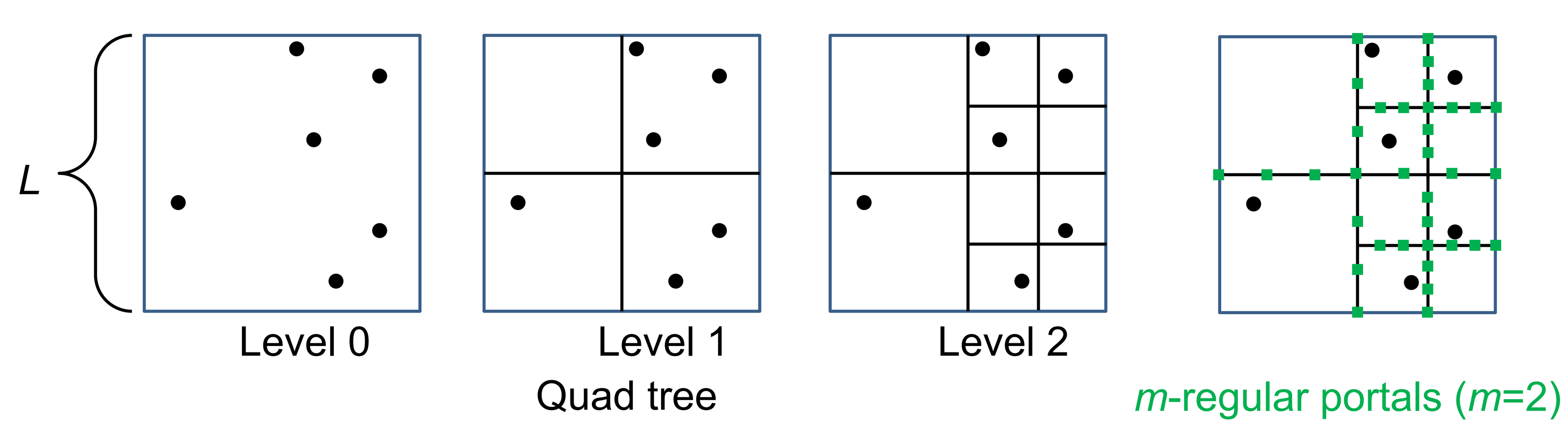
Rectilinear Steiner Minimum Trees

Problem Statement:
Input: Point set $V \subset \mathbb{R}^2$
Output: Tree T with vertices $U = V \cup S$ and minimum length under ℓ_1 distance



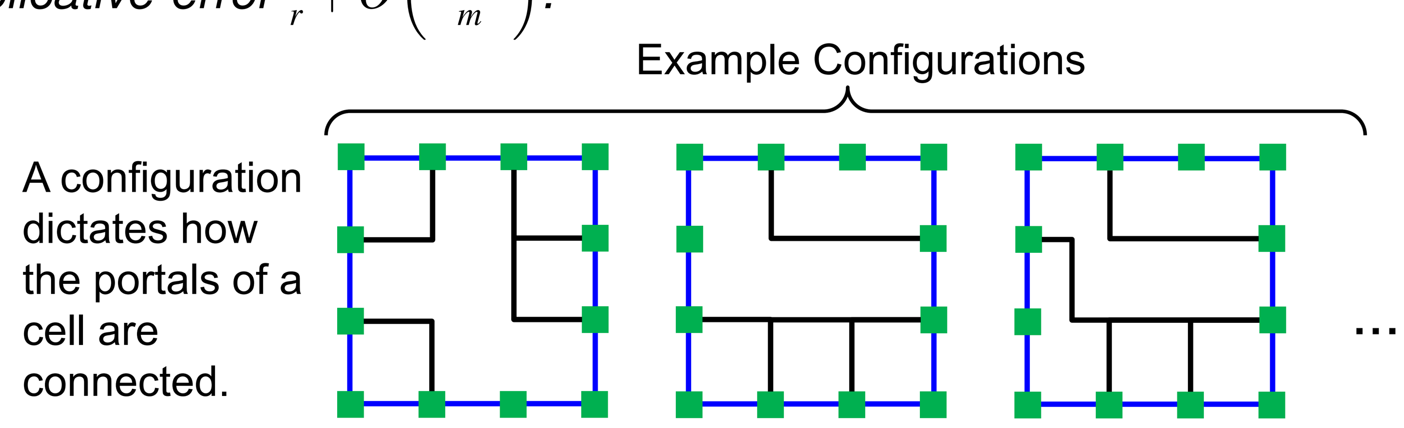
Arora's Algorithm

Key components:



Definition. *A tree is (m,r)-light if it crosses each side of each quad-tree cell at most r times, always at an m-regular portal.*

Theorem. *(m,r)-light trees approximate the length of RSMTs to within multiplicative-error $\frac{4}{r} + O\left(\frac{4logL}{m}\right)$.*



Algorithm [1]:

1. Construct quad tree
2. Base step: compute cost for each configuration of each quad-tree leaf
3. Dynamic programming step: compute the configuration costs for each quad-tree cell using the costs of its child cells
4. Combine costs at the quad-tree root to find the minimum-cost (m,r)-light tree

Problem: Number of configurations is bounded, but too large in practice
Solution:

- Keep the DP framework
- Replace brute-force computation of poartal configuration costs with neural networks

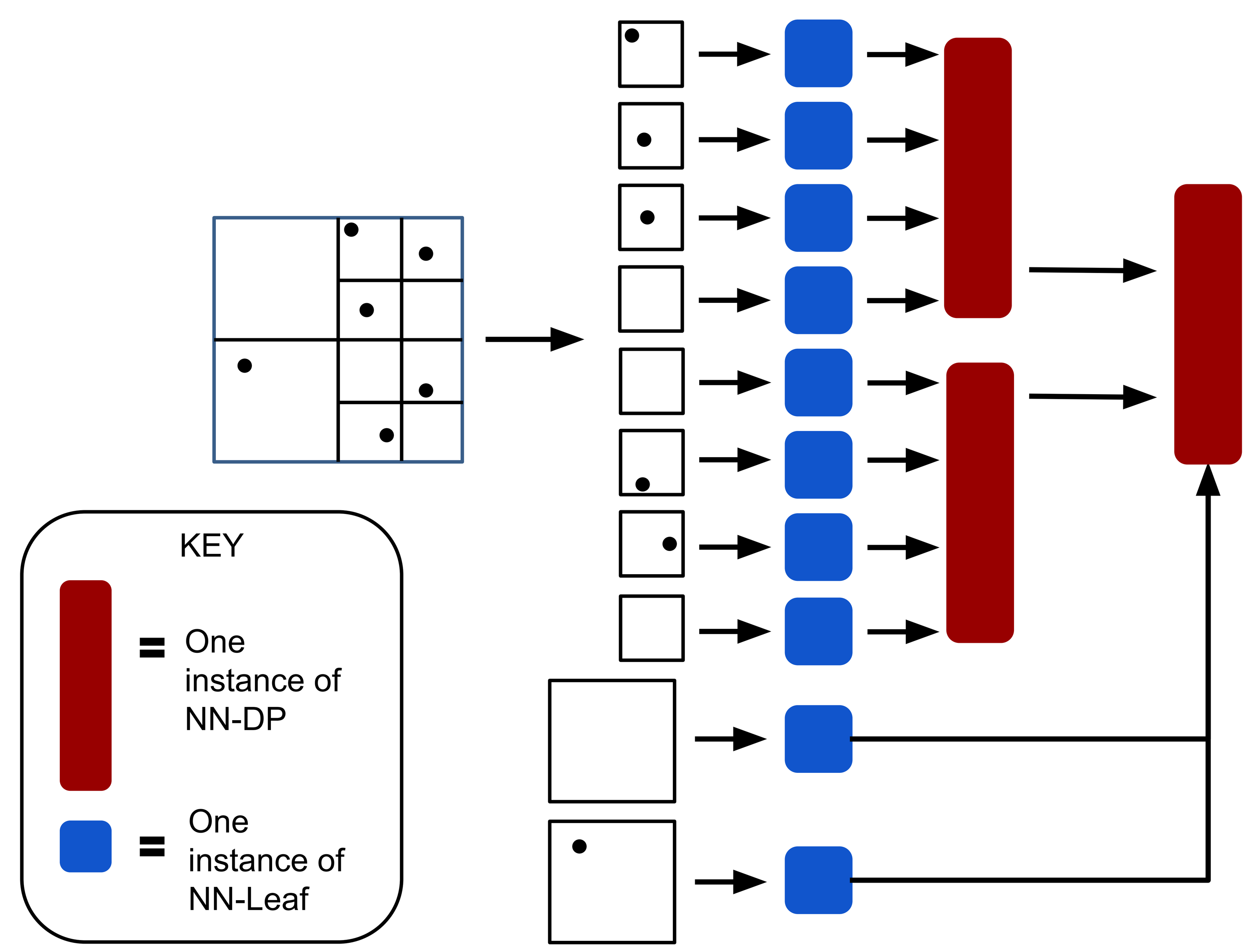
NN-Steiner

NN-Steiner is a mixed neural-algorithmic approach based on Arora's SOTA algorithm

Four bounded-size NN components, each called multiple times, are used:

- **NN-Leaf:** a cell's terminal and portal locations $\rightarrow$ an encoding of the cell's configuration costs
- **NN-DP:** output of 4 instances of NN-Leaf or NN-DP $\rightarrow$ an encoding of the cell's configuration costs
- **NN-Top:** output of top-level NN-DP $\rightarrow$ portal likelihoods
- **NN-Retrieve:** output of NN-DP and edge portal likelihoods $\rightarrow$ portal likelihoods

Thresholding the portal likelihoods at $t = .95$ yields the set of Steiner points S.



- NN components do not depend on problem size
- NN-Steiner generalizes to different problem sizes
- We can restrict training to fixed-sized problems!

Experimental Results

Algorithm \ Num. Points	50	100	200	500	800	1000	2000	5000
NN-Steiner	2.10	1.38	0.74	**-0.67**	**-1.11**	**-1.43**	**-2.44**	**-2.99**
REST [4]	**-0.17**	1.07	7.40	22.67	35.16	42.52		
FLUTE [2]	0.00	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.00
Geosteiner [3] (exact)	-0.55	-1.23	-2.25	-3.71	-4.43	-4.78		

Results are an average of 100 point sets sampled from a uniform distribution, and are reported as a percentage length-difference compared to FLUTE. REST is the SOTA NN algorithm.

Results show **NN-Steiner generalizes to large point sets**, despite training on point sets of size 180.

Acknowledgments

This work is partially supported by NSF under grants CCF-2112665, CCF-2217058 and CCF-2310411, and by DARPA IDEA HR0011-18-2-0032. The authors thank Anastasios Sidiropoulos for helpful discussions during initial stages of this project. We also thank Qi Zhao for early explorations and inputs to the early writing of this work.

References

[1] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[2] C. Chu and Y.-C. Wong. FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):70–83, 2007.

[3] D. Juhl, D. M. Warme, P. Winter, and M. Zachariasen. The GeoSteiner software package for computing Steiner trees in the plane: an updated computational study. *Mathematical Programming Computation*, 10(4):487–532, 2018.

[4] J. Liu, G. Chen, and E. F. Young. REST: Constructing rectilinear Steiner minimum tree via reinforcement learning. In *Proccedings of the 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1135–1140, 2021.