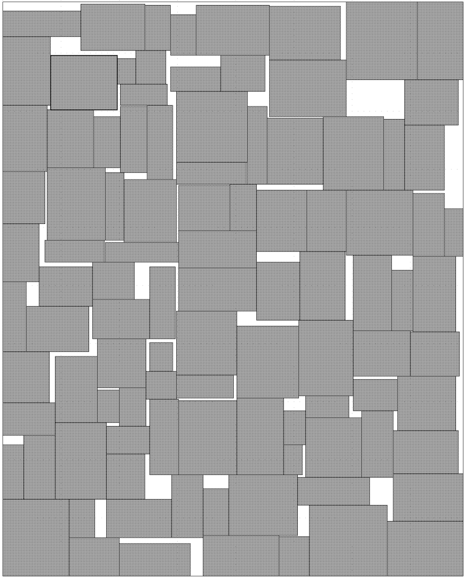
Learn to Floorplan through Acquisition of Effective Local Search Heuristics

Zhuolun He[1], Yuzhe Ma[1], Lu Zhang[1], Peiyu Liao[1], Ngai Wong[2], Bei Yu[1], Martin D.F. Wong[1]

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

Floorplanning: a Classic Problem

- Determine location of large blocks
- Minimize area, wirelength, ...
- Encode geometric relationship w/ dedicated data structure
- Perturb an encoding to generate new solutions (i.e., local search)



Enhancing the Local Search Method

How to enhance? Existing works adopt a local search algorithm, and

- Select a good start point [J.A. Boyan 98, Y. Zhou 16]
- Tune search parameters [U. Benlic 17]
- Scale the regularization term [D. Beloborodov 20]
- Switch between heuristics on the fly [A. Nareyek 03]

Our aim: acquire a local search algorithm from the scratch.

Intuition: prior human knowledge might mislead the learning!

Learning a Local Search Heuristic

We use *Reinforcement Learning* (RL) to learn a heuristic.

Problem Description

- An agent walks in the search space
- Selects a neighbor or rejects to move
- Gets to neighbor state or stays
- Optimize decision based on signals

Corresponding RL Formalization

- State s: a complete solution
- Action a: defined perturbations/reject
- Transition $\mathcal{T}$: (deterministic)
- Reward r: $\Delta cost$

Difficulties: large state/action space

Dealing with Large State Space: Neural Networks

- We use sequence pair [H. Murata 03] to encode floorplan.
- The state space is large: $(n!)^2 \times 8^n$ possible states for n blocks
  - two permutation in sequence pair
  - rotation/flip of blocks
- Tabular method infeasible
- Solution: utilize a neural network to approximate the policy
  - Input: features concatenated in 1D vector
  - Output: scalar for action value prediction
  - Architecture: a Multi-Layer Perceptron (MLP)

Dealing with Large Action Space: Sampling

- Possible actions: perturb sequence pair + rotate/flip blocks
- The action space is large: $\Theta(n^3)$ discrete actions for n blocks
  - space size is polynomial of problem size
  - action evaluation (state energy, action value) is costly
- Solution: sample actions in both training/testing
  - We proved its convergence in DQN.

Area and Wirelength Minimization on MCNC

- Baseline: carefully tuned Simulated Annealing (SA)
- Cost: $0.6 * area + 0.4 * wirelength$
- Our method outperforms in all cases: up to 2.5% lower cost and $5.7\times$ speedup

Circuit	Statistics		Area ($\times 10^6$)		WL ($\times 10^5$)		Cost ($\times 10^6$)		Runtime (s)	
	blocks	nets	Ours	SA	Ours	SA	Ours	SA	Ours	SA
apte	9	97	**47.08**	47.31	4.03	**3.43**	28.41	28.53	15.9	38.1
xerox	10	203	**20.42**	20.64	**6.33**	6.62	12.51	12.65	17.2	98.8
hp	11	83	**9.21**	9.40	**1.95**	2.62	5.60	5.74	11.6	44.3
ami33	33	123	**1.24**	1.25	0.69	**0.46**	0.77	0.77	43.1	82.2
ami49	49	408	**38.65**	39.47	**17.24**	12.31	23.88	24.18	66.8	165.0

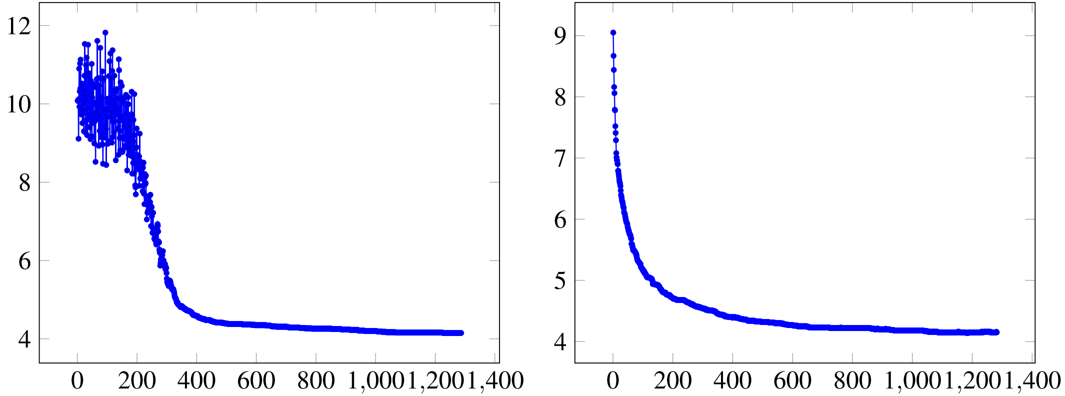
Area and Wirelength Minimization on GCRS

- Early stop criteria:
  - SA: no move accepted in the last 20 temperatures
  - Our agent: no better solution found in the last 100 steps
- Our method shows comparable performance, yet runs slower in larger instance

Circuit	Statistics		Area ($\times 10^5$)		WL ($\times 10^5$)		Cost ($\times 10^5$)		Runtime (s)	
	blocks	nets	Ours	SA	Ours	SA	Ours	SA	Ours	SA
n100	100	576	**1.95**	1.97	1.55	**1.54**	1.79	1.80	389.4	396.2
n200	200	1274	2.15	**2.01**	3.48	**3.34**	2.68	2.54	784.9	1101.9
n300	300	1632	3.40	**3.29**	**5.25**	5.44	4.14	4.15	3766.9	2062.3

Search Progress Visualization

- Interpretation: smoother $\Longrightarrow$ more greedy and deterministic

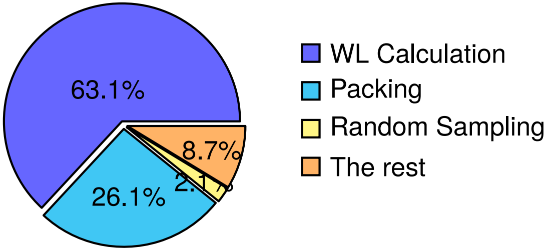


(a) Simulated Annealing

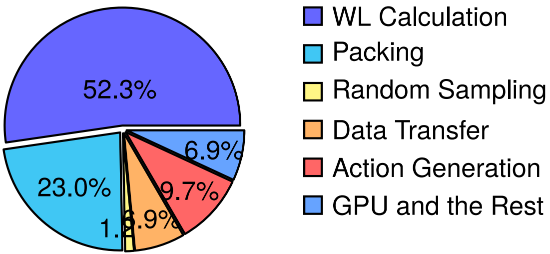
(b) Our agent

Runtime Profiling

- Platform: Python, CPU+GPU
- Solution evaluation takes most (89.2% and 75.3%) of the runtime.
- Random sampling only takes a little portion of time (2.1% and 1.2%).



(a) Simulated Annealing



(b) Our agent

Conclusion

- Problem: floorplaning
- Motivation: ‘learn’ new algorithms without human expert knowledge
- Formulation: local search, selecting neighbors
- Method: Deep Q-learning w/ action sampling
- We expect more research along this line!

Thank You