Machine Learning for
Combinatorial Optimization
COMPETITION 2021

NIPS 2021 ML4CO Dual Task 1st Solution

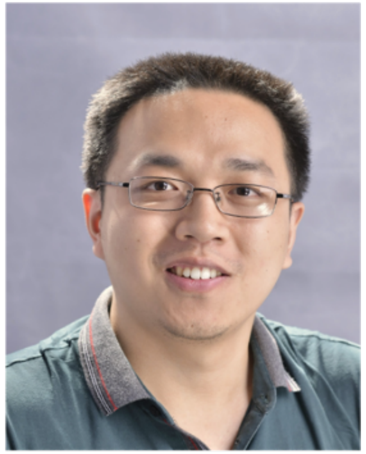
MEGVII 旷视 北京大学 PEKING UNIVERSITY

Zixuan Cao

Yang Xu

Zhewei Huang

Shuchang Zhou



Results

Dual task

team	item_placement			load_balancing			anonymous		
	dual integral	(cum. reward)	rank	dual integral	(cum. reward)	rank	dual integral	(cum. reward)	rank
Nuri	2307.39	(6684.00)	1	6178.82	(630787.18)	6	3770677.01	(27810782.42)	1
EI-OROAS	2321.09	(6670.30)	2	5221.69	(631744.31)	1	4423016.69	(27158442.74)	4
EFPP	2503.86	(6487.53)	3	5600.98	(631365.02)	3	5241194.97	(26340264.47)	13
KAIST_OSI	2794.83	(6196.56)	7	5555.42	(631410.58)	2	4955048.57	(26626410.86)	9
qqy	2614.16	(6377.23)	6	6408.69	(630557.31)	11	4359960.40	(27221499.03)	2
DaShun	4172.32	(4819.07)	14	6067.75	(630898.25)	4	4430033.29	(27151426.15)	5
lxj24	2547.84	(6443.55)	4	6812.76	(630153.24)	19	4529137.95	(27052321.48)	7
null_	4326.88	(4664.51)	16	6667.65	(630298.35)	14	4397369.92	(27184089.51)	3
Superfly	2967.19	(6024.20)	9	6219.04	(630746.96)	9	5208108.44	(26373350.99)	10

	Item Placement	Load Balancing	Anonymous
Baseline	4937.8	624043.6	30965031.6
Our Model	**7561.6 (1st)**	**624928.9 (6th)**	**32898846.0 (1st)**

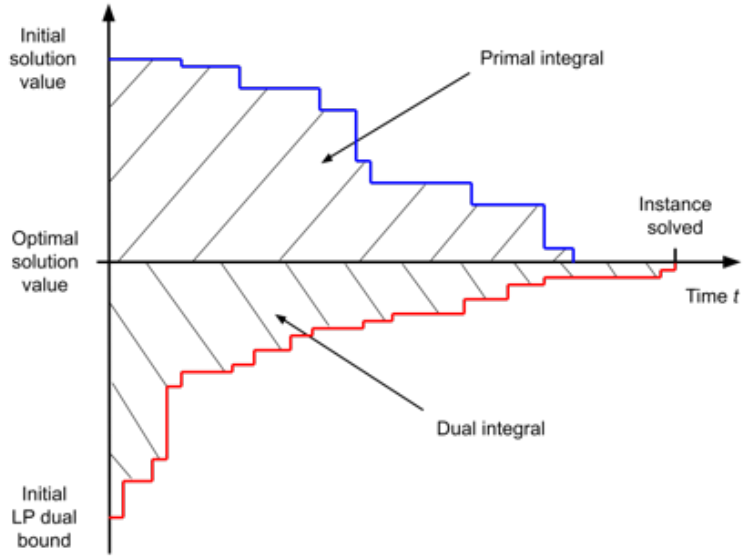
Performance Comparison on Validation Dataset

Overview

- Background
- Different Methods on Benchmarks
  - "Anonymous"
  - "Item Placement"
  - "Load Balancing"

Background: Dual Task VS Primal Task

- Primal Task: Produce feasible solutions, in order to minimize the **primal integral** over time
- Dual Task: Select branching variables, in order to minimize the **dual integral** over time
- Making branching decisions has received little theoretical understanding to this day

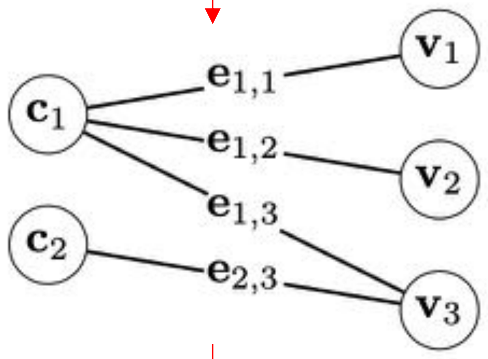


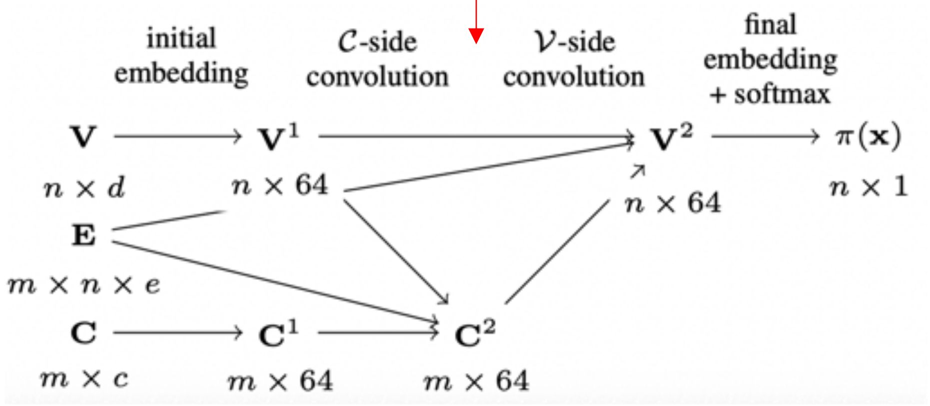
Background: Baseline Method

- Data Collection
  - Interact with CO solver SCIP and use Strong Branching as expert knowledge
- Training
  - Convert CO expressions to bipartite graphs
  - Parametrize variable selection policy as GCNN
  - Calculate cross entropy loss between GNN’ s node selection policy and expert knowledge of Strong Branching

X1 + X2 + X3 <= 5

X3 <= 2

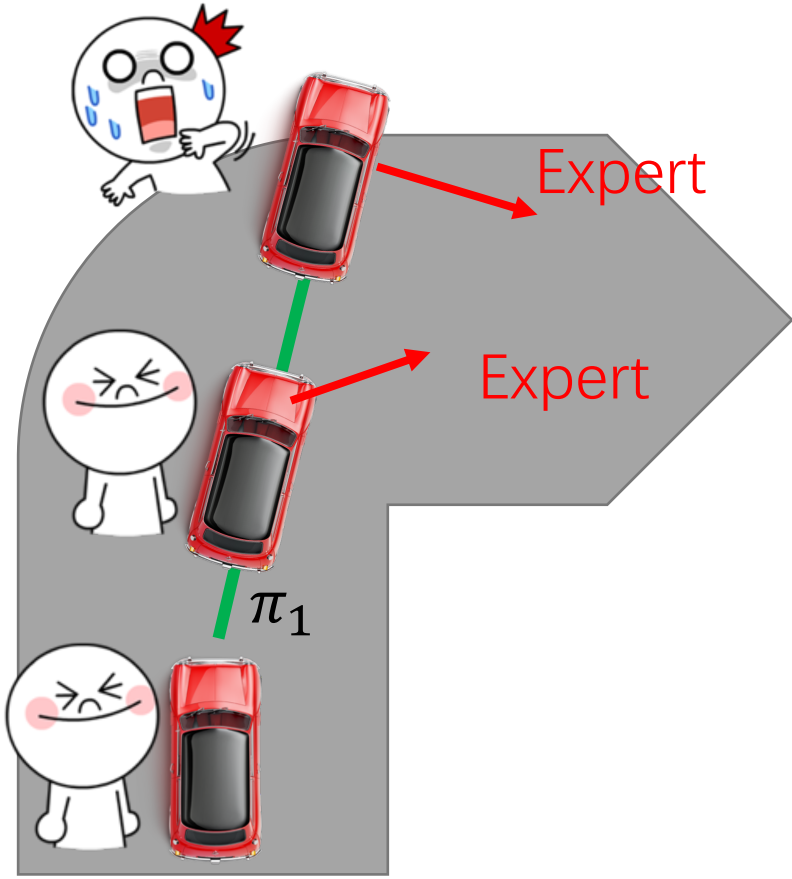




Exact Combinatorial Optimization with Graph Convolutional Neural Networks (NIPS 2019)

Improvement on "Anonymous"

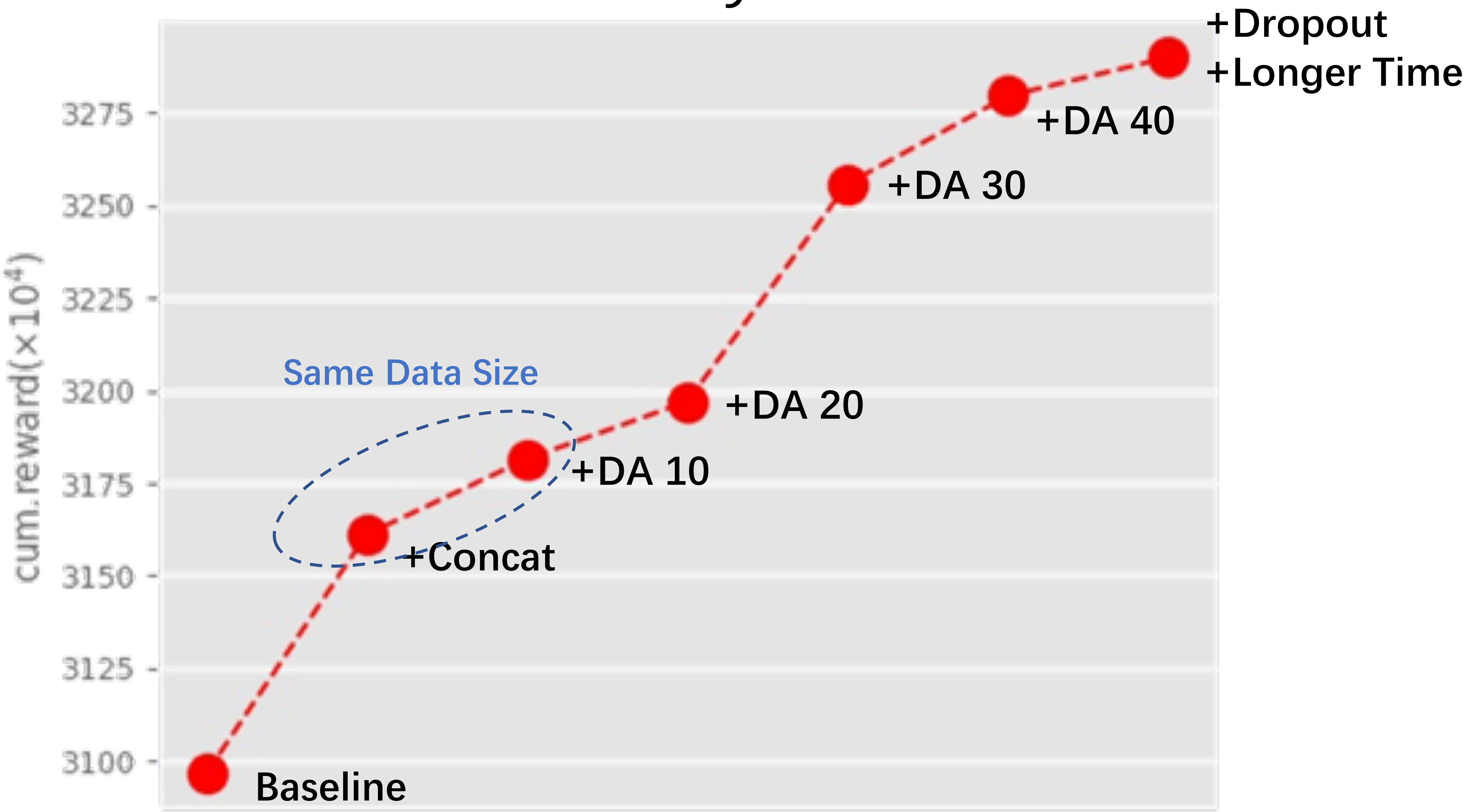
- Built upon Baseline Method
- Training only on "expert" data loses diversity
  - Use DAgger [1]
  - Initialize a random model π_0 and an empty dataset D
  - Iteratively collect new data and train new models
- Extra Tricks
  - Longer time limit when collecting: 15min → 20min
  - Summation to Concatenation:
    - R = f(g(L+R+Edge), R) → R = f(g(concat(L, R, Edge)),R)
  - Dropout
    - Add dropout layers (p=0.2) at embedding layers and final output layer



Demonstration of Dataset Aggregation [2]

[1] A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning (AISTATS 2011)
[2] Imitation Learning (Hongyi Li, Machine Learning 2021 Spring Course)

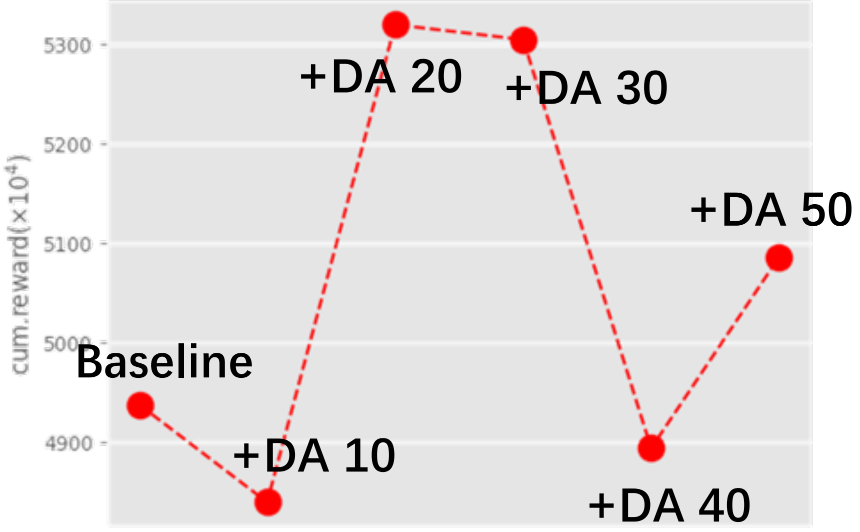
Improvement on "Anonymous"



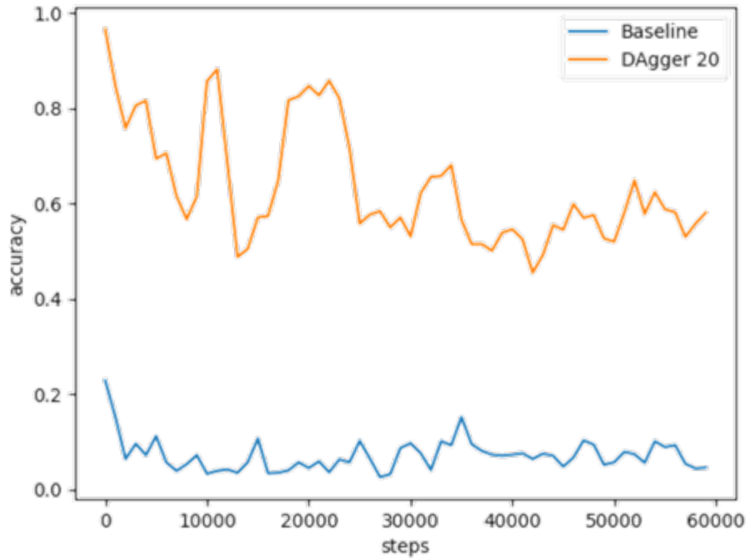
Cum. Reward of Anonymous in Validation Dataset

But, "Item Placement" is Different...

- DAgger improves accuracy **significantly**, but only improves rewards **slightly**



Cum. Reward of Item Placement in Validation Dataset

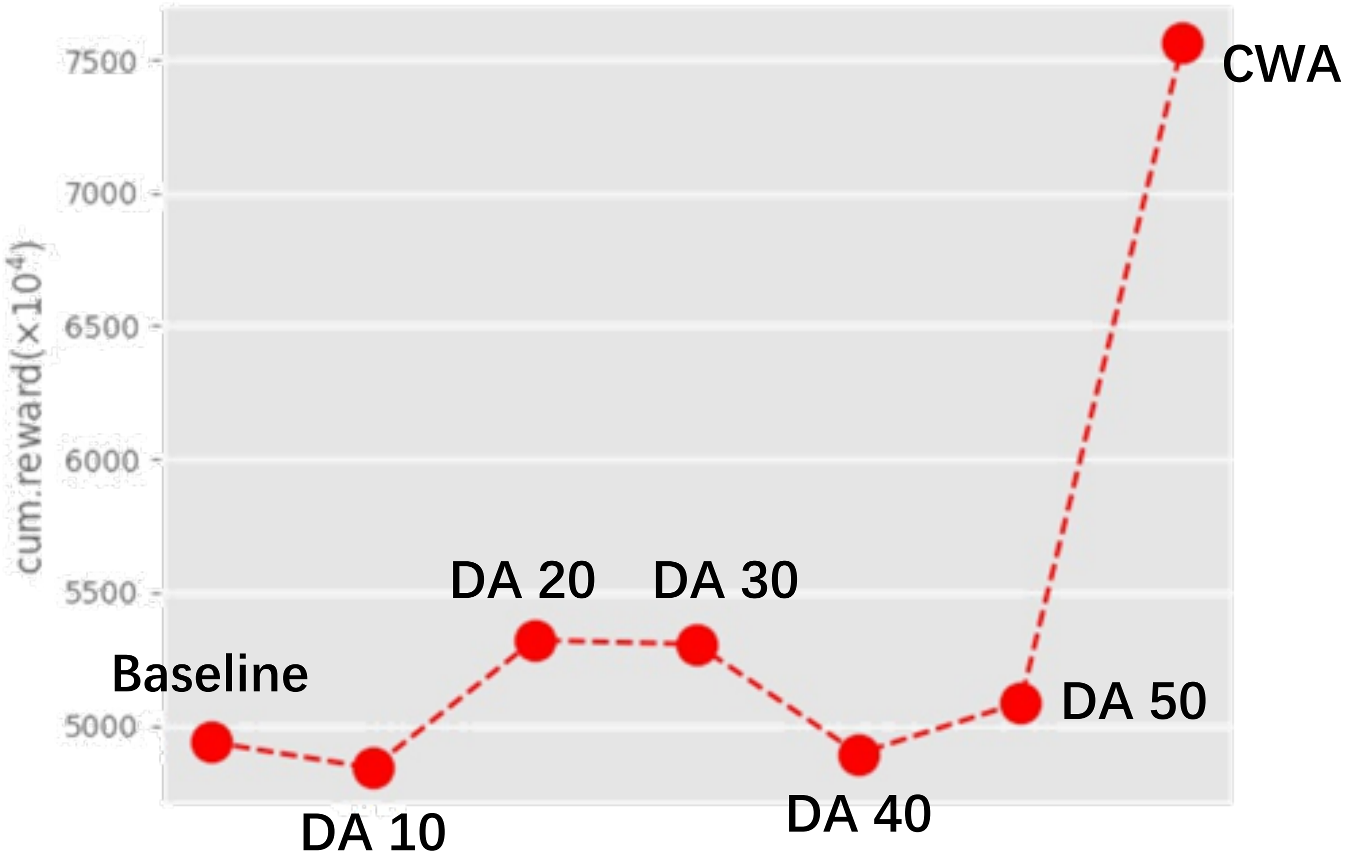


Accuracy When Interacting With Solver

	Accuracy When Interacting	Cum. Reward
Baseline	0.072	4937.8
DAgger 20	**0.635**	**5319.8**

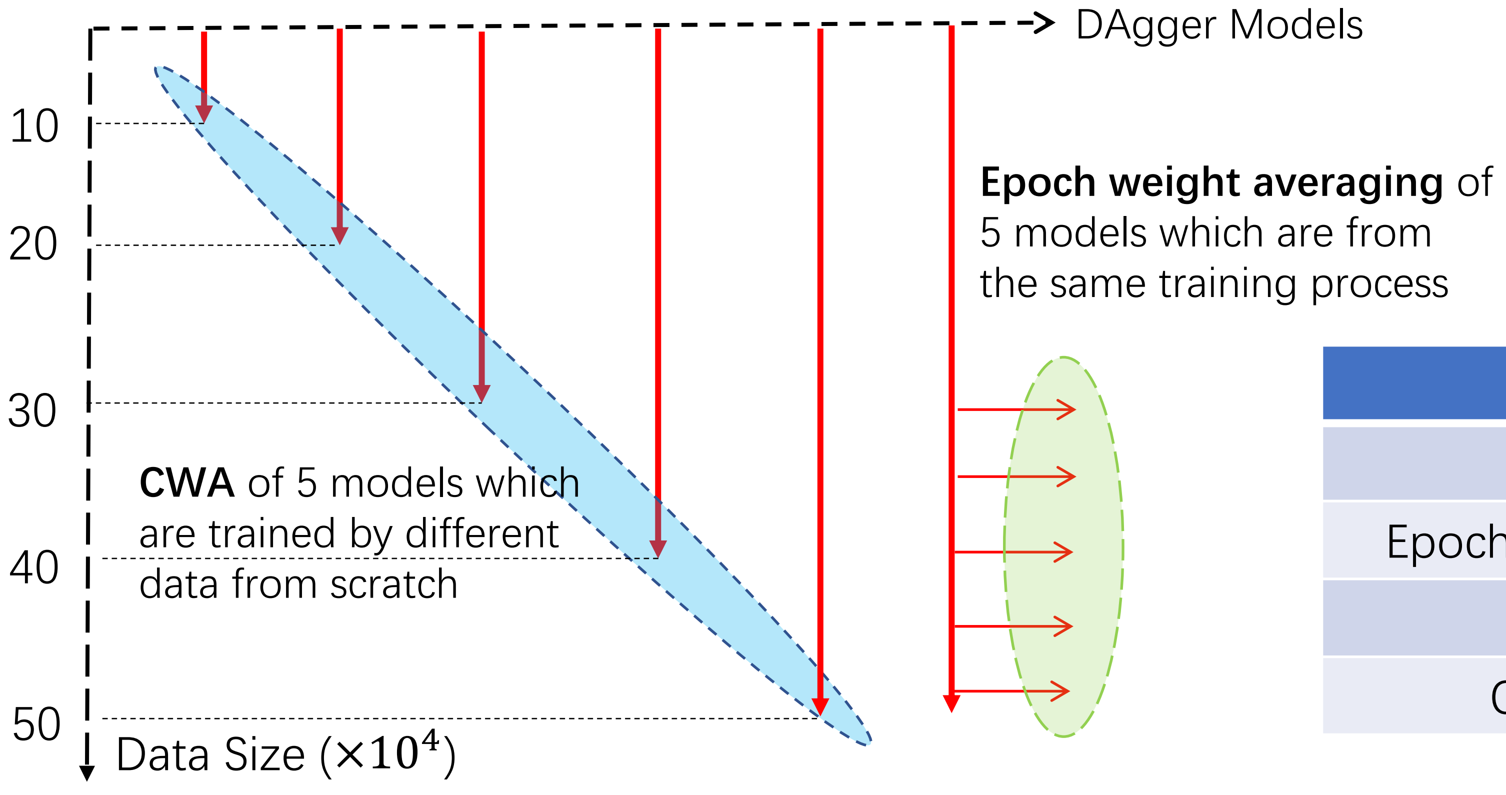
CWA: Cross-Model Weight Average

- CWA performs far more better than all previous trained models



CWA: Cross-Model Weight Averaging

- CWA: For models trained by different data from scratch with parameters $(\theta_0, \theta_1, \ldots, \theta_{n-1})$, build a new model π_{avg} with parameters $\theta_{avg} = \sum_{i=0}^{n-1} \theta_i / n$



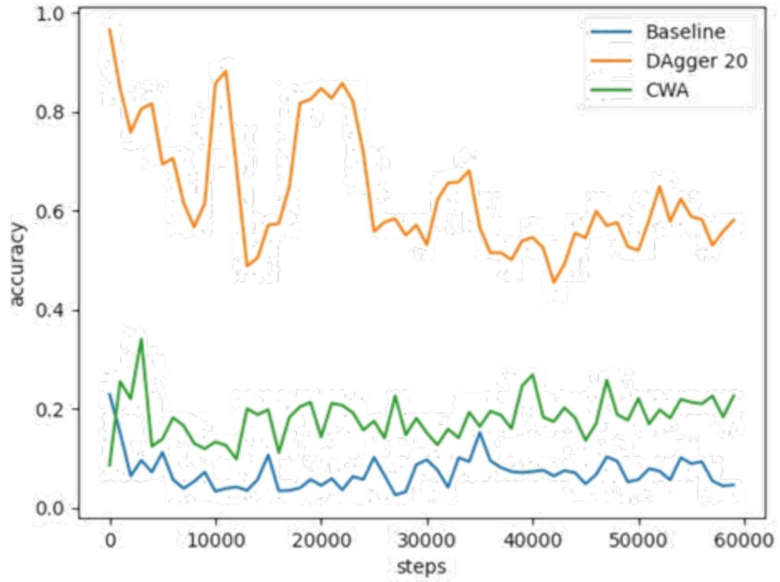
CWA VS Epoch Weight Average

	Cum. Reward
Baseline	4937.8
Epoch Weight Average	4759.1
CWA (5)	6850.7
CWA (Top 3)	**7561.6**

CWA: Cross-Model Weight Averaging

- CWA improves accuracy **slightly**, but improves rewards **significantly**

	Accuracy When Interacting	Cum. Reward
Baseline	0.072	4937.8
DAgger 20	**0.635**	5319.8
CWA (Top 3)	0.182	**7561.6**



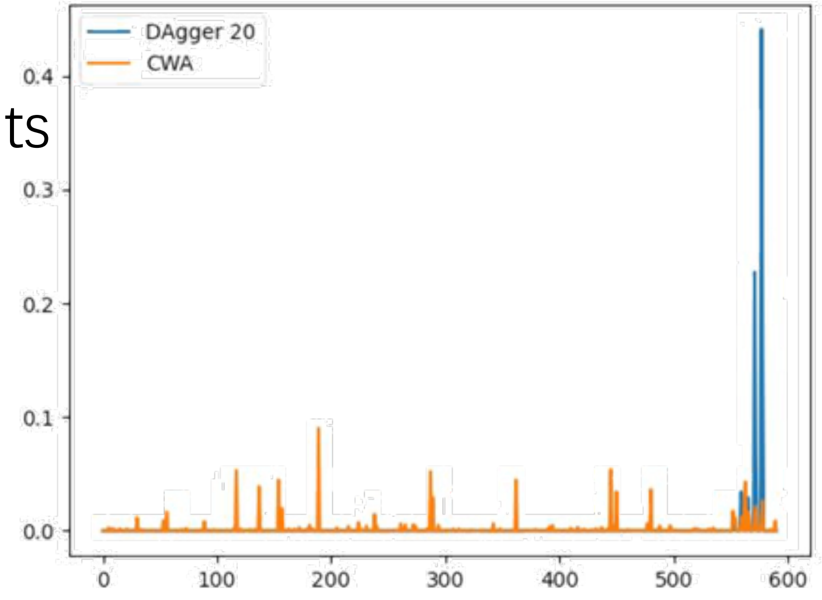
Accuracy When Interacting With Solver

CWA Works, but Why?

- ***We don' t know (yet)...***
- Test CWA on unseen dataset (collected by baseline method)
  - has lower accuracy but still has lower loss
  - produces more high probability selectable points

	Top 1	Top 3	Top 5	Loss	Cum. Reward
Model 0	**0.850**	**0.957**	**0.981**	7.38	5304.9
Model 1	0.797	0.917	0.966	9.50	5319.8
Model 2	0.795	0.916	0.961	6.18	5237.5
CWA	0.721	0.822	0.870	**2.96**	**7561.6**

Performance of models before and after weight averaging



Probabilities of Selecting Different Variables

Strong Branching Fails on "Load Balancing"?

- Models trained by baseline method and its modifications can not surpass random strategy.
- Strong Branching seems to fail

	Top 1	Top 3	Top 5	Cum. Reward
Baseline	**0.456**	**0.729**	**0.820**	624043.6
Random	0.013	0.034	0.052	**624928.9**

Performance Comparison of Baseline and Random

Conclusion

- Best performing methods have diverged on different benchmarks
    - "Anonymous": DAgger without CWA works
    - "Item Placement": DAgger with CWA rocks
    - "Load Balancing": Random strategy is the best we can get