COLLEGE OF COMPUTING AND INFORMATICS

1. RLScheduler: An Automated HPC Batch Job Scheduler Using Reinforcement Learning
2. SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning

Presenter: Yihe Wang
Department of Computer Science
University of North Carolina at Charlotte
ywang145@uncc.edu

COLLEGE OF COMPUTING AND INFORMATICS

RLScheduler: An Automated HPC Batch Job Scheduler Using Reinforcement Learning

- Introduction
  - Existing HPC batch job schedulers typically leverage heuristic priority functions to prioritize and schedule jobs. (ex. FCFS, SJF)
  - They are fixed and cannot automatically adapt to the variations in the target environment.
  - Ideally, an RL-based job scheduler will adapt to the varying job load as RL can continuously learn from trial-and-error as the load varies.

COLLEGE OF COMPUTING AND INFORMATICS

Background

Job Attributes

TABLE I: Description of job attributes.

Name	Symbol	Description
Job ID	id_t	the id of job
User ID	u_t	the user's ID
Group ID	g_t	the group's ID
Executable Id	app_t	ID of the job's executable file
Submit Time	s_t	job submission time
Requested Processors	n_t	the number of processors that a job requests.
Requested Time	r_t	job's runtime estimation (or upper bound) from users
Requested Memory	m_t	the requested memory per processor

Reinforcement Learning Framework

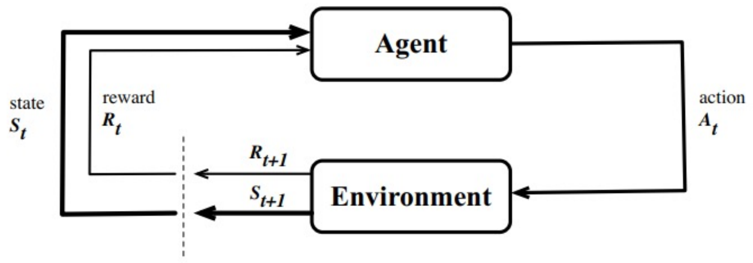


Fig. 1: General framework of reinforcement learning.

Background

- Scheduling Goal
  - Minimize the average waiting time (wait).
  - Minimize the average bounded slowdown (bsld).
  - Maximize resource utilization (util)

Question from Tanusri: How is backfilling being set?

- Scheduling and Backfilling
  - When a job waiting until its request to be satisfied, the backfilling can be activated to search for the jobs whose resource allocations can be satisfied now without affecting the planned execution for the waiting job

COLLEGE OF COMPUTING AND INFORMATICS

Discussion on Challenges

1) Take the waiting jobs and idle compute resources of the target HPC environment as the input for a deep neural network (DNN), 2)Use the DDN as the current scheduling policy to select a 'best' job as the action ,3)Apply the action back to the environment.

1. Job Order Change

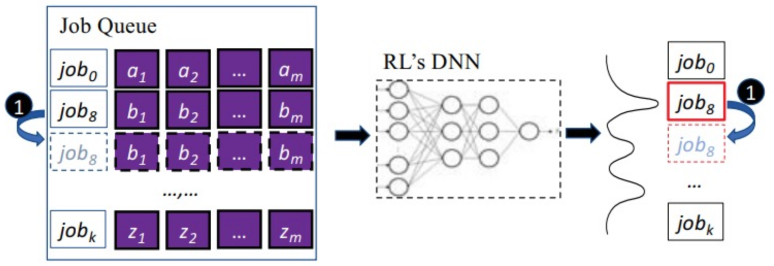

Job Queue
job_0 a_1 a_2 ... a_m
job_8 b_1 b_2 ... b_m
job_8 b_1 b_2 ... b_m
...,...
job_k z_1 z_2 ... z_m
RL's DNN
job_0
job_8
job_8
...
job_k
1
1

Fig. 2: The DNN-based RL agent reads waiting jobs and selects a job as the scheduling decision.

2. High variances in Samples

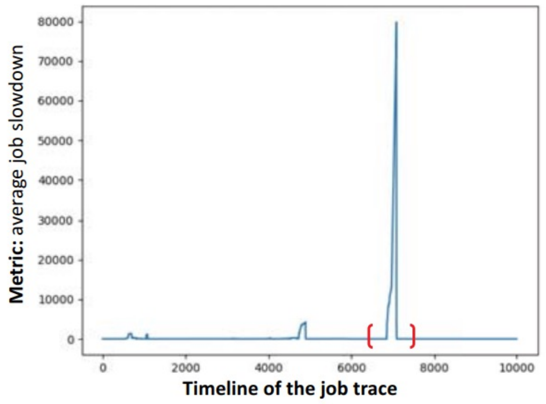


Fig. 3: The *average bounded slowdown* of scheduling sequence of 256 jobs in PIK-IPLEX-2009 job trace.

COLLEGE OF COMPUTING AND INFORMATICS

The overall architecture of RLScheduler

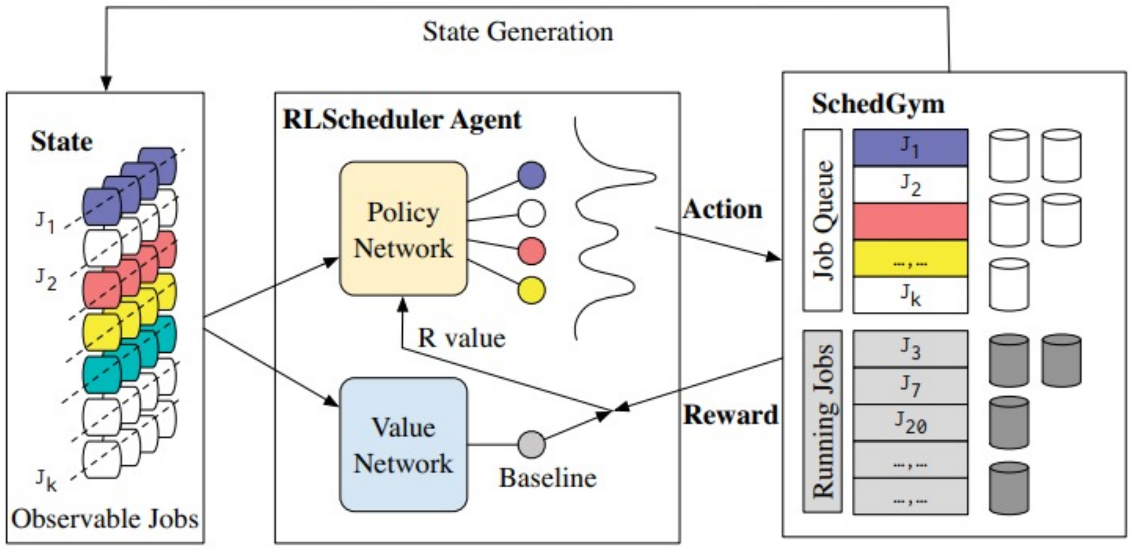


Fig. 4: The overall architecture of RLScheduler.

COLLEGE OF COMPUTING AND INFORMATICS

Policy Network

Kernel-based Neural Network

- For each waiting job, the network outputs a value, a calculated ‘score’ of the job. The values of all waiting jobs form a vector.
- Once jobs are reordered, their probabilities will also be reordered accordingly

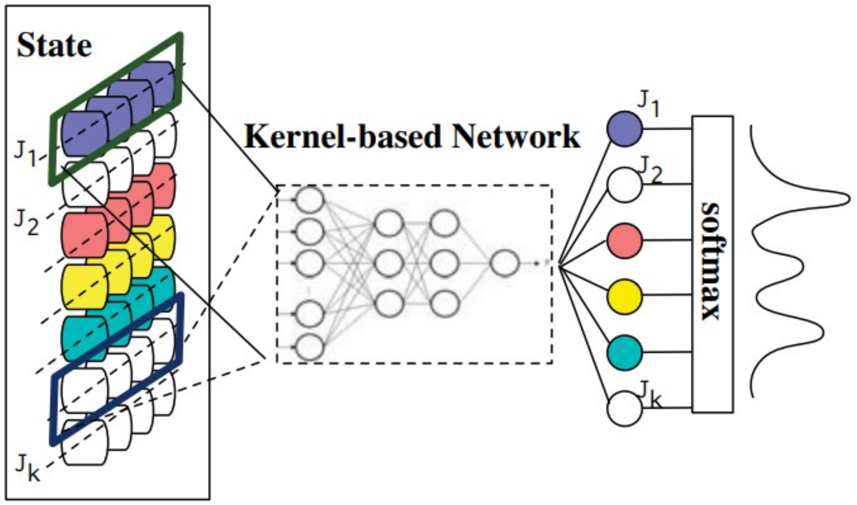


Fig. 5: The RLScheduler policy network structure. Its core is a kernel-based neural network.

COLLEGE OF COMPUTING AND INFORMATICS

Value Network

Question from Tanusri: SchedInspector uses the Actor-Critic model to accelerate and stabilize the training. Why does the training need to be stabilized and why is the Actor-Critic model the best model?

Compute the reward

- For a sequence of jobs, after the policy network makes all the scheduling decisions, we collect the rewards r.
- The output of value network can be intuitively considered as the expected reward (exp_r). Use (r−exp_r) to train the policy.
- This difference can be intuitively considered as the **improvement of current policy over historical policies on this set of jobs.**

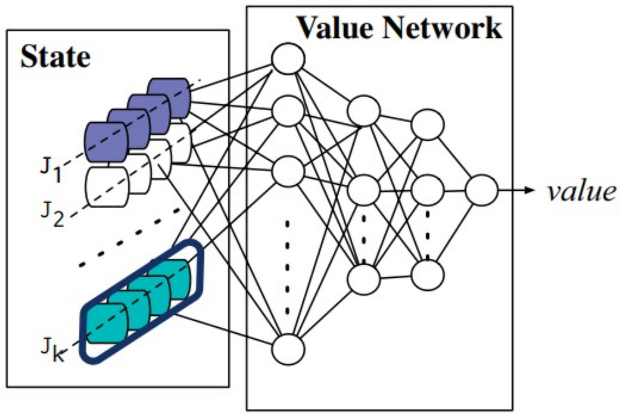


Fig. 6: The RLScheduler value network structure. Its core is a 3-layer multiple layer perceptron network (MLP).

COLLEGE OF COMPUTING AND INFORMATICS

Variance Reduction

Trajectory filtering – Solve the High variances in Samples

- It filters the 'easy sequences' out since they will not contribute info to improve the RL agent. For the 'non-easy sequences', it categorizes all sequences into two ranges and trains the RL agent in two steps.
  - 1) The first step contains job sequences whose variances fall into a specific range (R).
  - 2) The second step trains on all the job sequences

Question from Hrushi: How does the system handle variations in job requests and resource availability over time?

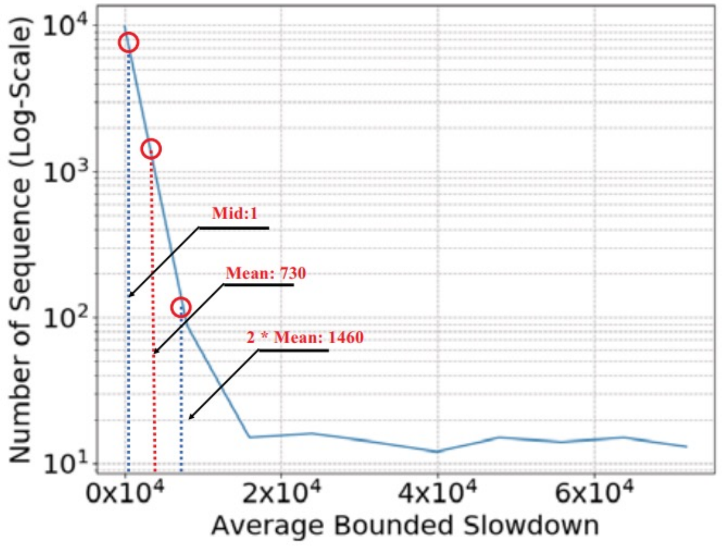


Fig. 7: The distribution of the *average bounded slowdown* of scheduling sequence of 256 jobs in PIK-IPLEX-2009 job trace.

COLLEGE OF COMPUTING AND INFORMATICS

Evaluation

- Address three questions
  - Whether the new designs (kernel-based neural network and trajectory filtering mechanism) improve the training performance of RLScheduler?
  - How well are RLScheduler's training and performance towards: different HPC workloads, different scheduling metrics, or even combined scheduling metrics?
  - Will a scheduling policy that RLScheduler learns still be applicable to an unseen, new workload?

COLLEGE OF COMPUTING AND INFORMATICS

Kernel-based Neural Network Performance

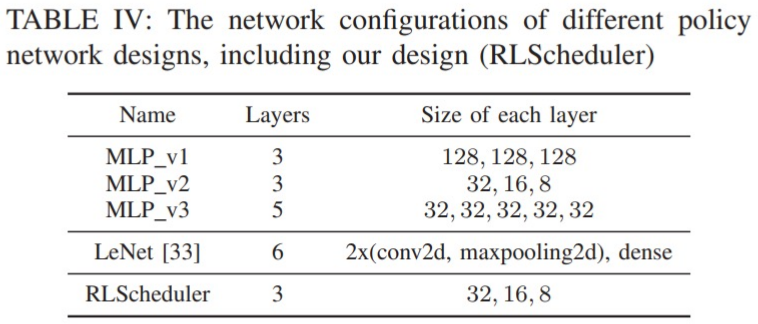
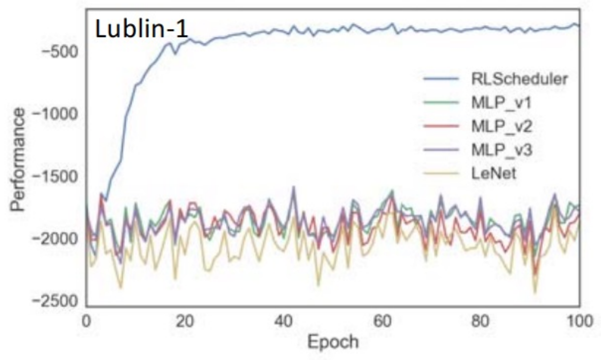
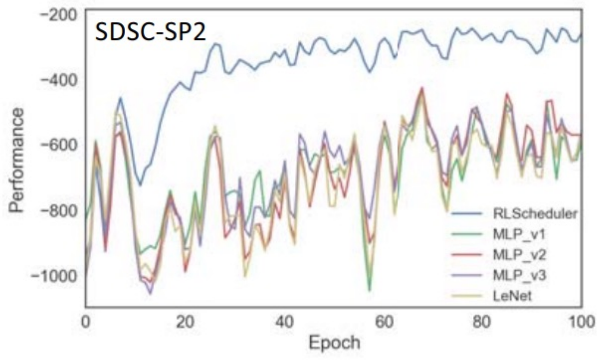


TABLE IV: The network configurations of different policy network designs, including our design (RLScheduler)

Name	Layers	Size of each layer
MLP_v1	3	128, 128, 128
MLP_v2	3	32, 16, 8
MLP_v3	5	32, 32, 32, 32, 32
LeNet [33]	6	2x(conv2d, maxpooling2d), dense
RLScheduler	3	32, 16, 8





- Question from Chris: How might the trade-off between size of the policy network and its resultant change in computational overhead impact the performance of the SchedInspector?
  - More parameters do not necessarily mean better performance.

COLLEGE OF COMPUTING AND INFORMATICS

Trajectory Filtering Performance

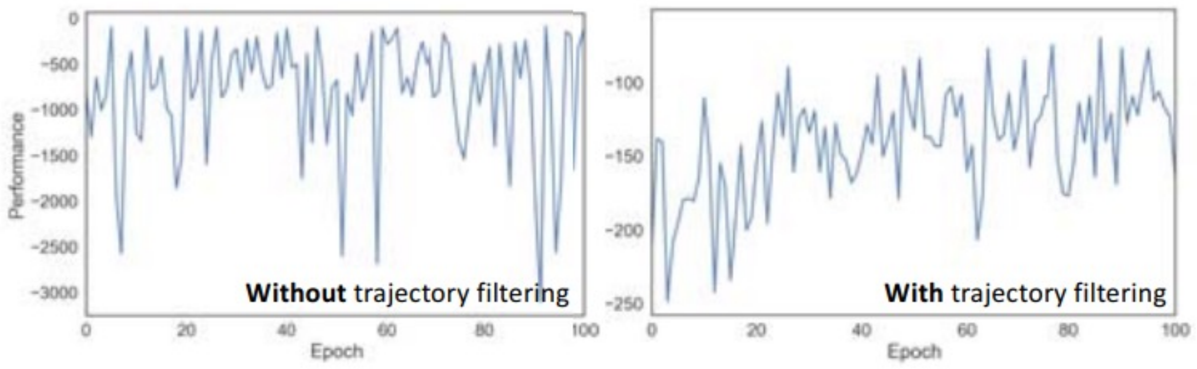


Fig. 9: The training curves of RLScheduler on PIK-IPLEX-2009 job trace *with* and *without* trajectory filtering.

COLLEGE OF COMPUTING AND INFORMATICS

Different Workloads and Goals

Bounded Slowdown

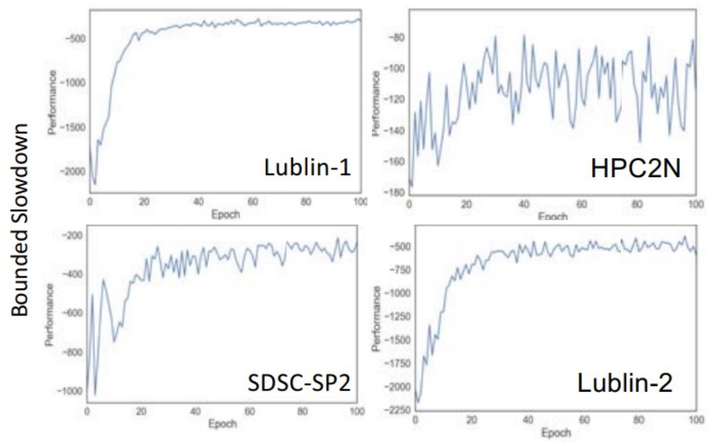


Fig. 10: The training curves of RLScheduler on four different workloads. *Here, all vertical axis are the average bounded slowdown calculated after scheduling the whole epoch. The horizontal axis shows the training epoch.*

Resource utilization

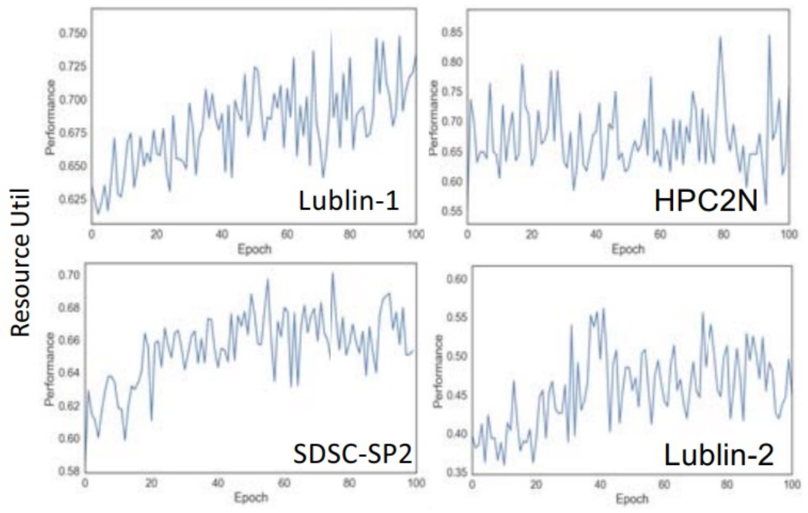


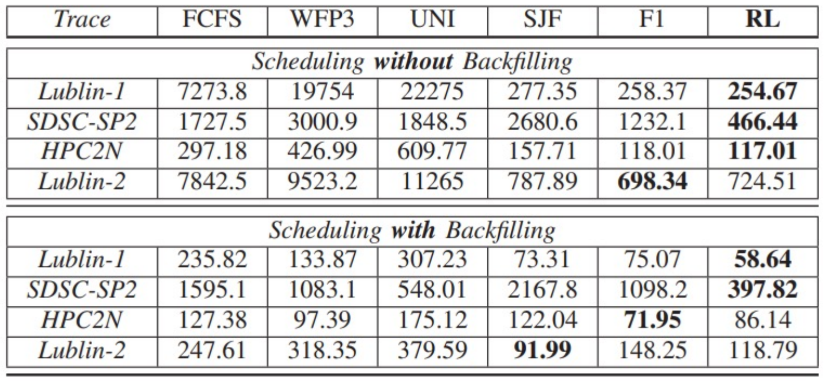
Fig. 11: The training curves of RLScheduler in four different job traces targeting *resource utilization*.

COLLEGE OF COMPUTING AND INFORMATICS

Different Workloads and Goals

Bounded Slowdown

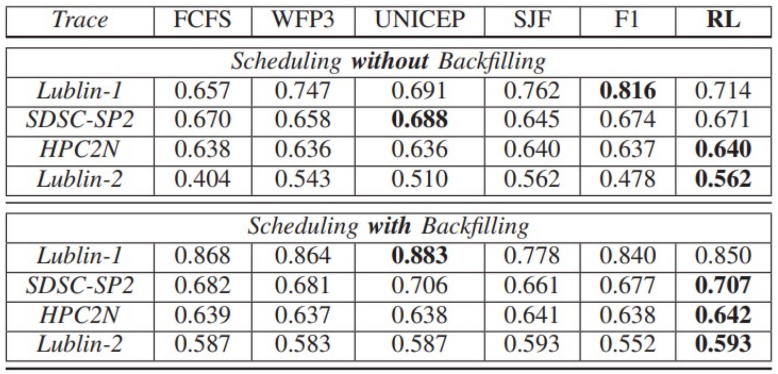
TABLE V: Results of scheduling different job traces.



Trace	FCFS	WFP3	UNI	SJF	F1	**RL**
*Scheduling **without** Backfilling*						
Lublin-1	7273.8	19754	22275	277.35	258.37	**254.67**
SDSC-SP2	1727.5	3000.9	1848.5	2680.6	1232.1	**466.44**
HPC2N	297.18	426.99	609.77	157.71	118.01	**117.01**
Lublin-2	7842.5	9523.2	11265	787.89	**698.34**	724.51
*Scheduling **with** Backfilling*						
Lublin-1	235.82	133.87	307.23	73.31	75.07	**58.64**
SDSC-SP2	1595.1	1083.1	548.01	2167.8	1098.2	**397.82**
HPC2N	127.38	97.39	175.12	122.04	**71.95**	86.14
Lublin-2	247.61	318.35	379.59	**91.99**	148.25	118.79

Resource utilization

TABLE VI: Results of scheduling towards resource utilization.



Trace	FCFS	WFP3	UNICEP	SJF	F1	**RL**
*Scheduling **without** Backfilling*						
Lublin-1	0.657	0.747	0.691	0.762	**0.816**	0.714
SDSC-SP2	0.670	0.658	**0.688**	0.645	0.674	0.671
HPC2N	0.638	0.636	0.636	0.640	0.637	**0.640**
Lublin-2	0.404	0.543	0.510	0.562	0.478	**0.562**
*Scheduling **with** Backfilling*						
Lublin-1	0.868	0.864	**0.883**	0.778	0.840	0.850
SDSC-SP2	0.682	0.681	0.706	0.661	0.677	**0.707**
HPC2N	0.639	0.637	0.638	0.641	0.638	**0.642**
Lublin-2	0.587	0.583	0.587	0.593	0.552	**0.593**

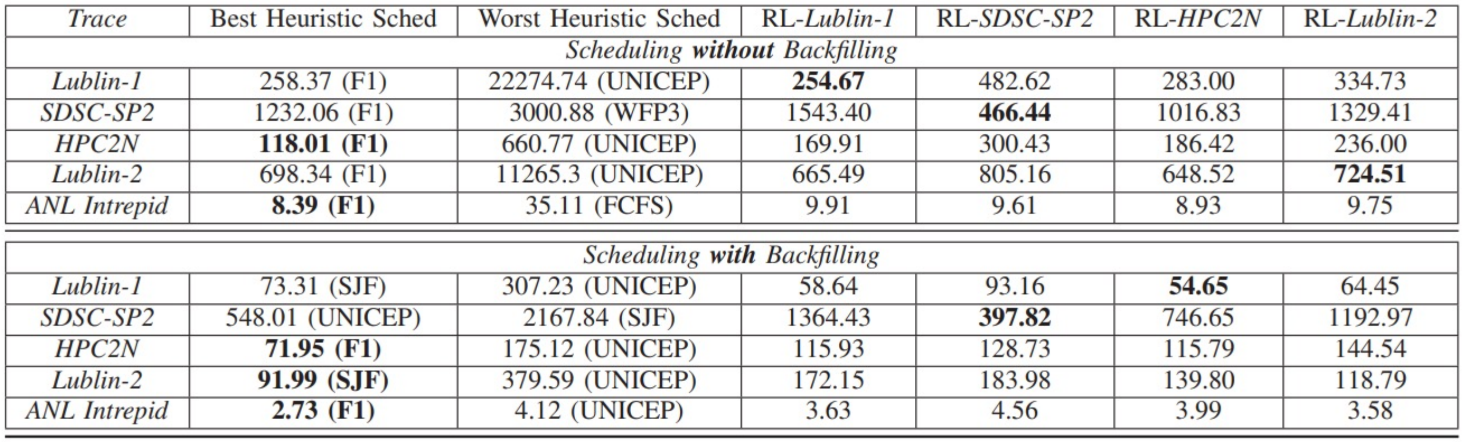
A heuristic scheduler that performs well on one goal may perform poorly on another goal even scheduling the same workload, while RLScheduler can adapt to different workloads and optimization goals with good performance

COLLEGE OF COMPUTING AND INFORMATICS

RLScheduler Stabilization

- Whether the learned model would be too specific to the given job trace and can not handle even small shifts in the workload?
  - Applying the learned RL model (RL-X) from job trace (X) onto other job traces (Y) and see how it would perform. It will be no worse than using an inappropriate heuristic scheduler.

TABLE VII: Performance comparisons of one RL-learned model (RL-X) being applied to other job traces (Y).



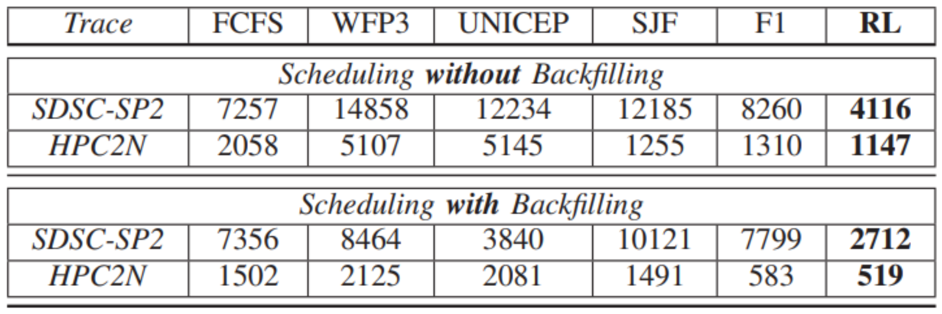
Trace	Best Heuristic Sched	Worst Heuristic Sched	RL-*Lublin-1*	RL-*SDSC-SP2*	RL-*HPC2N*	RL-*Lublin-2*
		*Scheduling **without** Backfilling*				
Lublin-1	258.37 (F1)	22274.74 (UNICEP)	**254.67**	482.62	283.00	334.73
SDSC-SP2	1232.06 (F1)	3000.88 (WFP3)	1543.40	**466.44**	1016.83	1329.41
HPC2N	**118.01 (F1)**	660.77 (UNICEP)	169.91	300.43	186.42	236.00
Lublin-2	698.34 (F1)	11265.3 (UNICEP)	665.49	805.16	648.52	**724.51**
ANL Intrepid	**8.39 (F1)**	35.11 (FCFS)	9.91	9.61	8.93	9.75
		*Scheduling **with** Backfilling*				
Lublin-1	73.31 (SJF)	307.23 (UNICEP)	58.64	93.16	**54.65**	64.45
SDSC-SP2	548.01 (UNICEP)	2167.84 (SJF)	1364.43	**397.82**	746.65	1192.97
HPC2N	**71.95 (F1)**	175.12 (UNICEP)	115.93	128.73	115.79	144.54
Lublin-2	**91.99 (SJF)**	379.59 (UNICEP)	172.15	183.98	139.80	118.79
ANL Intrepid	**2.73 (F1)**	4.12 (UNICEP)	3.63	4.56	3.99	3.58

COLLEGE OF COMPUTING AND INFORMATICS

RLScheduler with Fairness

- In real world, scheduler needs to consider not only the average slowdown of all jobs, but also the average slowdown of each user’s jobs.
- use Maximal as the aggregator, which means RLScheduler will focus on the user with maximal job slowdown and learn to prioritize the user to minimize the overall maximal.

TABLE VIII: Results of scheduling different job traces towards *bounded job slowdown* with *Maximal Fairness*.



Trace	FCFS	WFP3	UNICEP	SJF	F1	**RL**
*Scheduling **without** Backfilling*						
SDSC-SP2	7257	14858	12234	12185	8260	**4116**
HPC2N	2058	5107	5145	1255	1310	**1147**
*Scheduling **with** Backfilling*						
SDSC-SP2	7356	8464	3840	10121	7799	**2712**
HPC2N	1502	2125	2081	1491	583	**519**

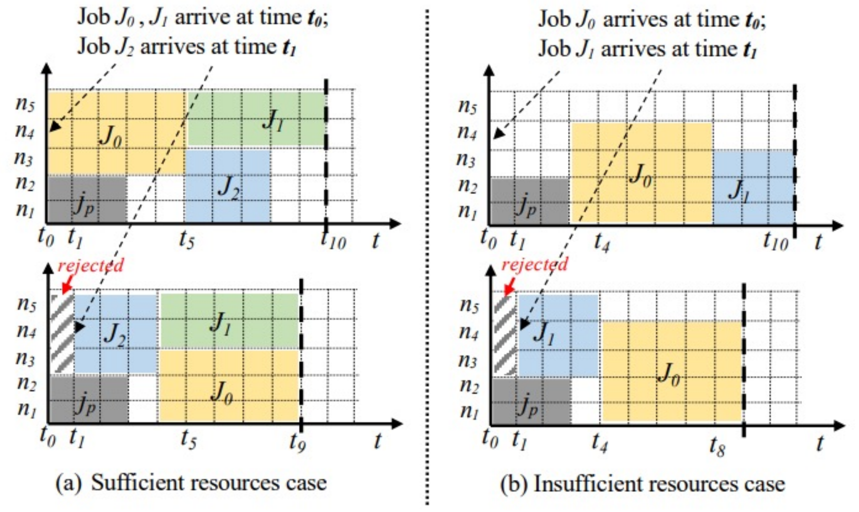
COLLEGE OF COMPUTING AND INFORMATICS

SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning

- This paper integrate runtime factor into existing batch job scheduling
  - This paper introduces a scheduling inspector to scrutinize the scheduling decisions made by the existing scheduling policy.
  - If it believes the current job as a good fit for the runtime, the scheduling continues as normal. Otherwise, this scheduling decision will be rejected and the job will be put back to the waiting queue and be considered again at the next scheduling point.

COLLEGE OF COMPUTING AND INFORMATICS

An Example of Better Case



Job J_0, J_1 arrive at time t_0; Job J_2 arrives at time t_1
Job J_0 arrives at time t_0; Job J_1 arrives at time t_1
rejected
rejected
(a) Sufficient resources case
(b) Insufficient resources case

Figure 1: Scheduling jobs with/without SchedInspector. *x-axis shows the timeline in minutes; y-axis shows the compute nodes ($n_{1\rightarrow5}$); each block represents a job that takes amount of nodes and time; J_p is the preliminary job running before the scheduling starts.*

- we notice that the better performance comes from the cases:
  - 1)new jobs arrived and were added into the waiting queue before the next scheduling point;
  - 2) the new jobs match the cluster availability better and are scheduled to improve the performance
- Although we cannot accurately predict the exact features or the arrivals of future jobs, from the historical job and environmental statistics, we still can learn when rejection has higher chance to win

COLLEGE OF COMPUTING AND INFORMATICS

The overall architecture of SchedInspector

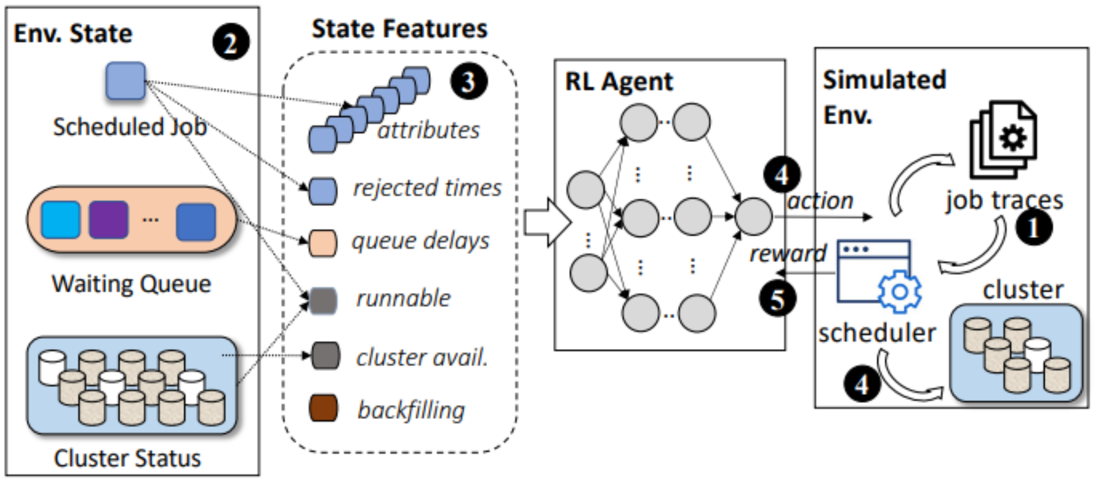


Figure 3: Architecture of SchedInspector.

The policy and value networks are the same in RL agent.

COLLEGE OF COMPUTING AND INFORMATICS

Feature Building

- Scheduled job
  - job waiting time ($wait$)
  - job execution time (est)
  - job requested computing nodes (res)
- Rejected times
  - Question from Tanusri: How is MAX_REJECTION_TIMES set? If it set arbitrarily will affect the training performance?
  - Hyperparameter: MAX_REJECTION_TIMES
- Queue delays
  - Iterate all of the waiting jobs, calculate their expected delays according to the given performance metrics, and add them together as the value of queue delays.
- Runnable and Cluster availability
  - ratio of free computing nodes(n_free) and total computing nodes (n_total) is the Cluster availability. Runnable value is 1 meaning the job can run immediately; otherwise, its value is 0.
- Backfilling Contributions
  - If it is enabled, we scan the waiting jobs and calculate the number of waiting jobs that can be backfilled as the final value of this feature.

COLLEGE OF COMPUTING AND INFORMATICS

Reward Function

Question from Trevon: What biases might the reward function introduce that lead to bad performances?
Question from Wes: Can you explain the reward function in more detail on how it works with the SchedInspector?

- Native reward
  - Average bounded slowdown($blsd$): reward = $bsld\_orig - bsld\_inspect$.
  - Drawbacks: The improvements of reward in a job sequence with large *bsld* can be easily larger than a job sequence with smaller *bsld,* which may confuse the RL agent during training.

- Win/Loss reward
  - $count(bsldinspect < bsldorig)$: It will not be affected by the variances of the metric values.
  - Drawbacks: it treats all improvements the same, hence does not reward the big-gain actions.

- Percentage Reward
  - $(bsld\_orig - bsld\_inspect)/bsld\_orig$: It does not have the previous two drawbacks.

COLLEGE OF COMPUTING AND INFORMATICS

Evaluation

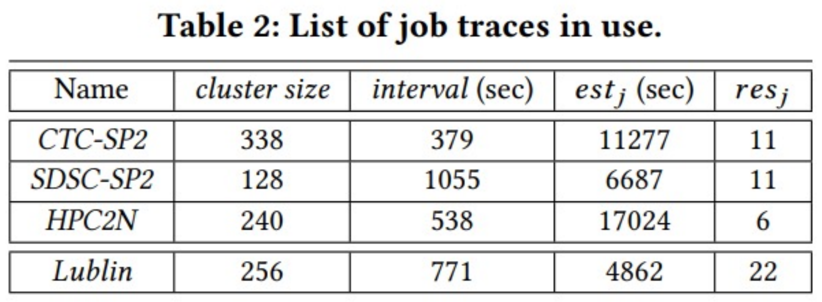


Table 2: List of job traces in use.

Name	*cluster size*	*interval* (sec)	est_j (sec)	res_j
CTC-SP2	338	379	11277	11
SDSC-SP2	128	1055	6687	11
HPC2N	240	538	17024	6
Lublin	256	771	4862	22

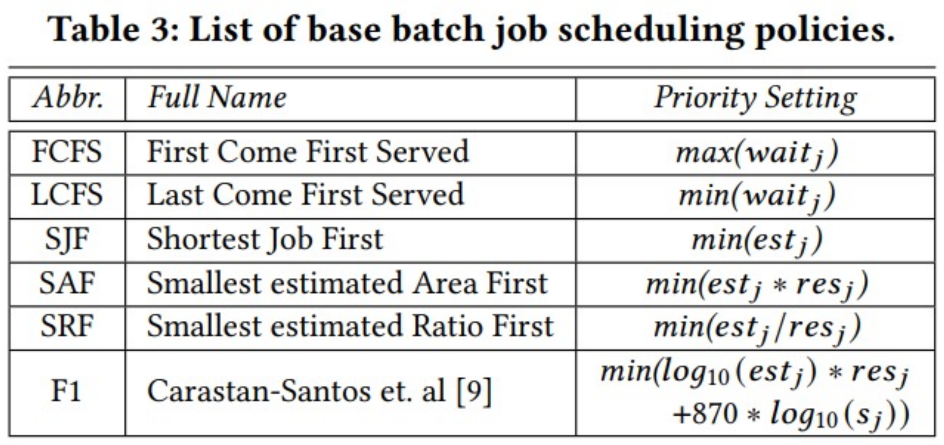


Table 3: List of base batch job scheduling policies.

Abbr.	*Full Name*	*Priority Setting*
FCFS	First Come First Served	$max(wait_j)$
LCFS	Last Come First Served	$min(wait_j)$
SJF	Shortest Job First	$min(est_j)$
SAF	Smallest estimated Area First	$min(est_j * res_j)$
SRF	Smallest estimated Ratio First	$min(est_j / res_j)$
F1	Carastan-Santos et. al [9]	$min(log_{10}(est_j) * res_j + 870 * log_{10}(s_j))$

Question from Shreya: How do you ensure that SchedInspector is not overfitting to th training data and is generalizing well to new, unseen data?

To avoid over-fitting, for each trace we use the first 20% for training and the remaining 80% of the data for testing.

COLLEGE OF COMPUTING AND INFORMATICS

Overall Performance

- 4 job traces
- Two policies: SJF and F1
- One goal: average bounded job slowdown (*bsld*)

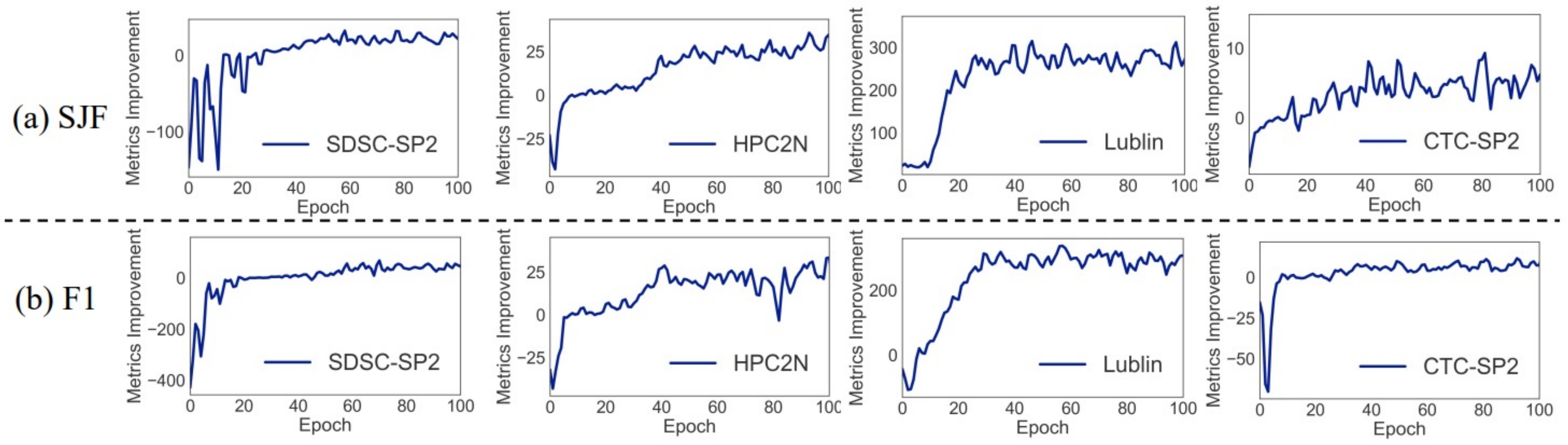


Figure 4: Training curves of SchedInspector on four job traces using two schedulers. ***The x-axis shows the training epoch, the y-axis shows the metrics improvements on the selected metrics ($bsld$). Larger than 0 means SchedInspector performs better than the base schedulers.***

COLLEGE OF COMPUTING AND INFORMATICS

Impacts of Feature Building

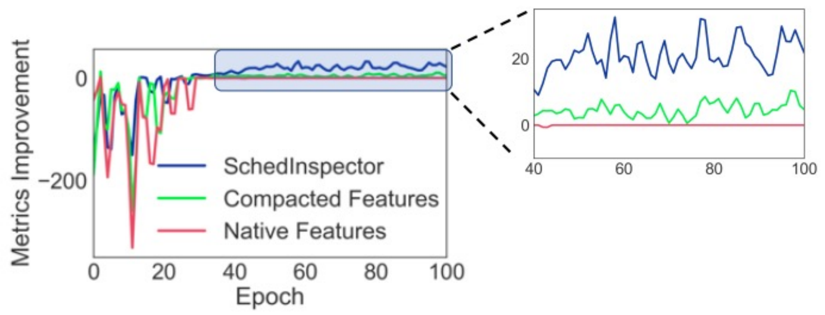


Figure 5: The comparison of the training curves of SchedInspector with different feature building mechanisms. *y-axis shows the improvements of SchedInspector over the base scheduling policy on* bsld*. Larger is better.*

- Native Features: it directly use the whole environmental state as the inputs.
- Compacted Features: it only the current job and the cluster state and ignores the queue delay and backfilling contributions.
- SchedInspector Features: Our case.

COLLEGE OF COMPUTING AND INFORMATICS

Impacts of Reward Function

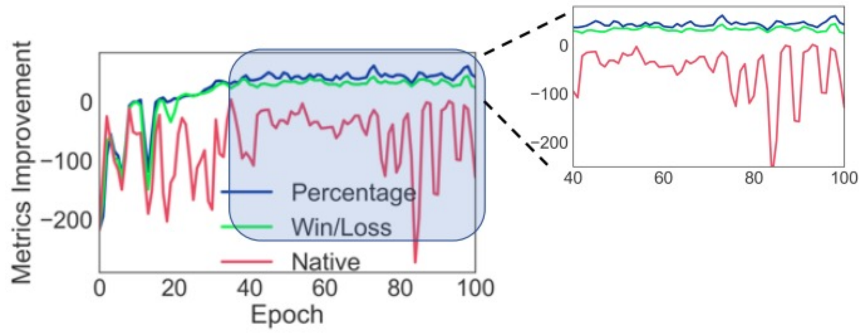


Figure 6: The comparison of the training curves of SchedInspector with different reward functions.

- Native reward
- Win/loss reward
- Percentage reward: it stabilizes the highly variant reward values as well as captures the big gains.

COLLEGE OF COMPUTING AND INFORMATICS

Working with Various Scheduling Policies

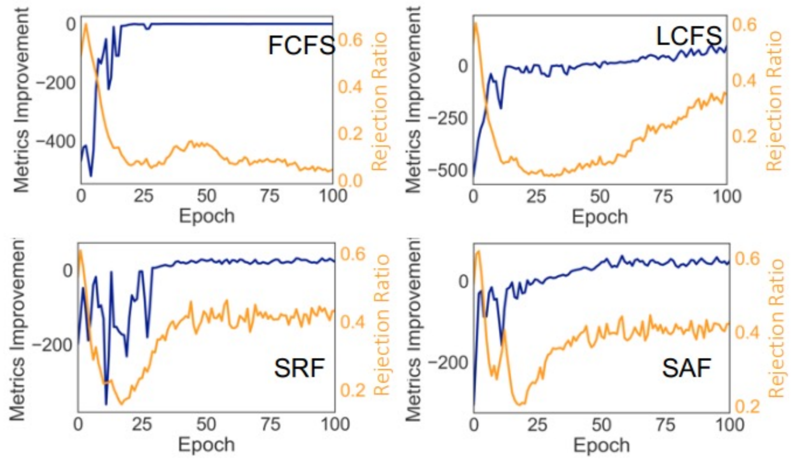


Figure 7: SchedInspector training with different base job scheduling policies. ***Blue curve is the** bsld **improvements using the left** y**-axis; orange curve is the rejection ratio using the right** y**-axis.***

- The key is whether rejecting current job can lead to a different job being scheduled in the future. If nothing changes after some idle time, then the rejection becomes a pure waste.
- Since FCFS always prioritizes the oldest job, any future job will not impact its decision.

COLLEGE OF COMPUTING AND INFORMATICS

Working with Different Job Traces

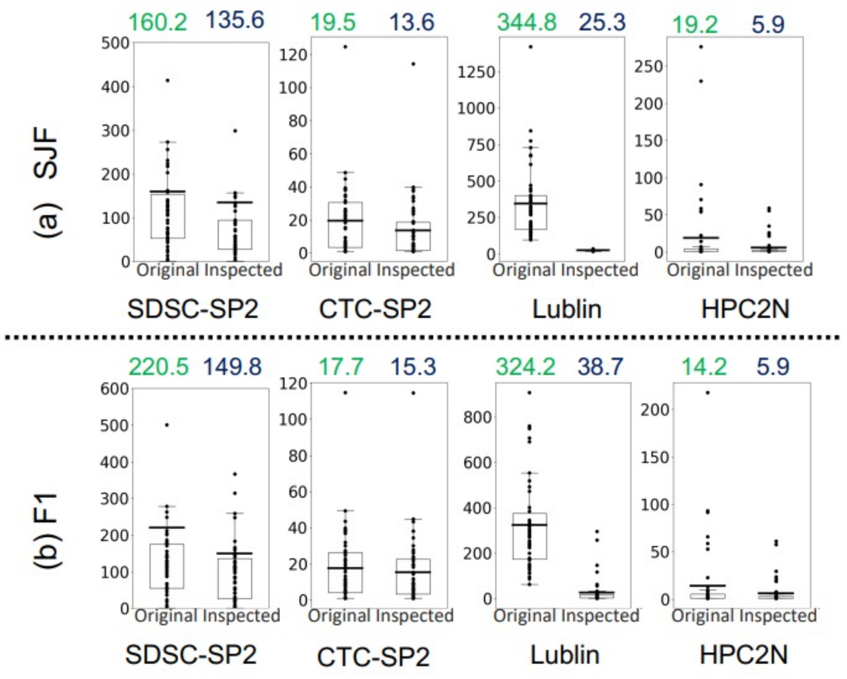


Figure 8: The scheduling performance of SchedInspector and base scheduling policies on four different job traces. ***y-axis is the bsld value. The averages are shown on the top of each bar. Smaller is better.***

COLLEGE OF COMPUTING AND INFORMATICS

Working with Different Job Execution Metrics

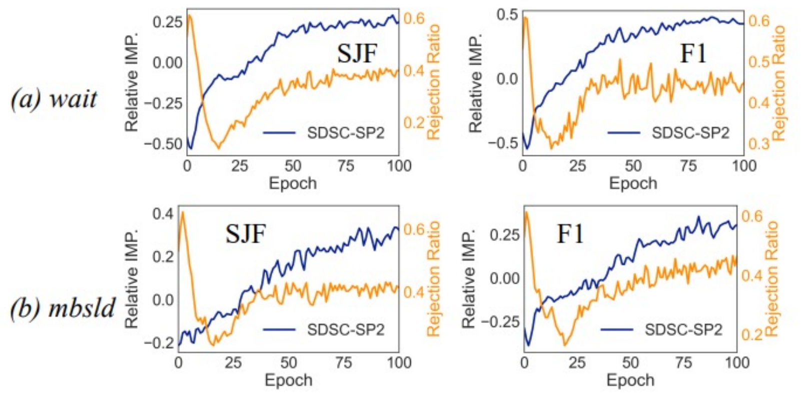
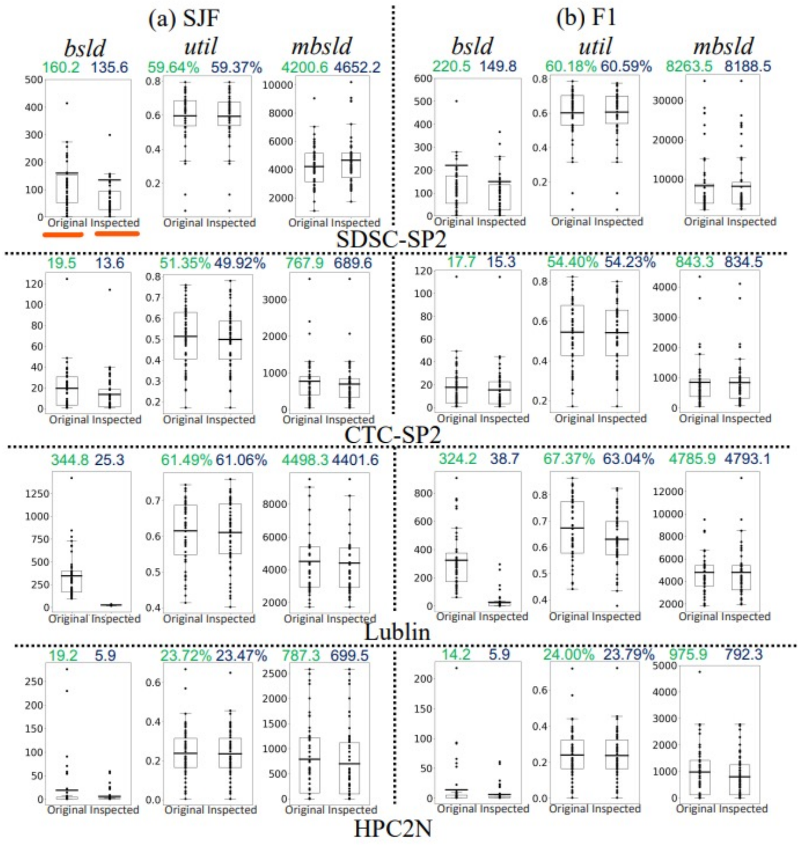


Figure 9: SchedInspector training with different job execution performance metrics. ***Blue curve is the relative improvement using the left y-axis; orange curve is the rejection ratio using the right y-axis.***

- Average waiting time (wait): the average duration between the job's submission and its start time. It does not consider job length in its calculation.
- Maximal bounded job slowdown ($mbsld$): the maximal $bsld$ of a job sequence instead of the average. It emphasizes on the fairness and effectively avoids starving long jobs.

COLLEGE OF COMPUTING AND INFORMATICS

Trade-off Among Different Metrics



- We trained SchedInspector towards *bsld* and evaluated it towards *util* and *mbs*ld.
- The result shows the rejections introduced by SchedInspector do not break the system utilization.

COLLEGE OF COMPUTING AND INFORMATICS

SchedInspector in Realistic Settings

Question from Atharva: In real-world circumstances, how effective is SchedInspector at increasing batch task scheduling performance?

$$\begin{aligned} Job\_Priority = &(weight_{age}) * (age\_factor) + \\ &(weight_{fairshare}) * (fairshare\_factor) + \\ &(weight_{jattr}) * (job\_attribute\_factor) + \\ &(weight_{partition}) * (partition\_factor) + \ldots \end{aligned}$$

- Real world batch job schedulers are more complicated.
- Slurm multifactor priority scheduler with backfilling as the base scheduling policy.

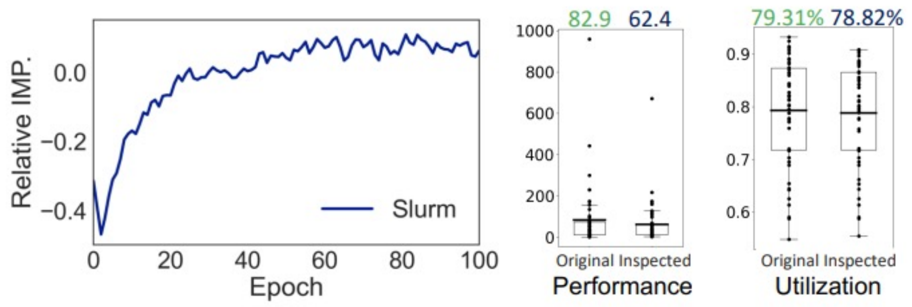


Figure 12: The performance of SchedInspector working with Slurm. ***y-axis on the right two charts are*** *bsld* ***and util percentage.***

Goal: bsld

COLLEGE OF COMPUTING AND INFORMATICS

Some questions at the end

- Question from Chris: Since the approach seems to hurt performance at the beginning of training, and the output is less interpretable than heuristic algorithms, is it likely that system administrators may be almost equally hesitant to implement such a tool as a fully new scheduling algorithm?
- Question from Jonathan: What are the advantages of SchedInspector over other approaches?
- Question from Uzochi: If one were to train a machine learning model to replace the existing job scheduling heuristic algorithms, would SchedInspector still be necessary?
  - Not necessary. The advantage of SchedInspector is it is more interpretable than pure RL agent.
- Question from Wes: How do you think the SchedInspector can be improved where and why?
  - Maybe do some online learning?
- What are the differences among these job traces? Do they cover all the situations in scheduling?
  - I don’t know actually.

COLLEGE OF COMPUTING AND INFORMATICS

Questions