An Index Advisor Using Deep Reinforcement Learning

Hai Lan[1], Zhifeng Bao[1], Yuwei Peng[2]

[1]RMIT University
[2]Wuhan University

Index Selection Problem (ISP)

Index Selection Problem (ISP)

- Choosing the right indexes to build is one of the central issues in database tuning.
- Problem Definition:
  - Select a set of indexes (index configuration) to be built to maximize the performance of the given workload with some constraints.
  - Constraints: storage usage, index number, and so on.

Index Selection Problem (ISP)

- Choosing the right indexes to build is one of the central issues in database tuning.
- Problem Definition:
  - Select a set of indexes (index configuration) to be built to maximize the performance of the given workload with some constraints.
  - Constraints: storage usage, index number, and so on.

- Index interaction: an interaction exists between an index a and an index b if the benefit of a is affected by the existence of b and vice-versa.

SELECT * FROM t WHERE a < 10 OR b < 10;

(1) An index on a ✗

(2) An index on b ✗

(3) An index on a and an index on b ✓

Prior Work

Prior Work

Category	Work	Cost	Index type	IIA	Alog	Cons
Non-Learning method	AutoAdmin [VLD'97]	Estimated cost	S/M	✗	Greedy	index number
	ILP [ICDE'07]	Estimated cost	S/M	✗	ILP	storage
	ISRM [ICDE 19]	Estimated cost	S/M	✓	Greedy	storage
Learning-based method	AI Meet AI [SIGMOD'19]	Learning-model Estimated cost	S/M	Not sure	Greedy	index number
	Welborn et al [arxiv'19]	Not mention	S/M	✓	DQN	no
	DRL-Index [ICDEW'20]	Estimated cost	S	✓	DQN	Not mention

IIA means index interaction. **Cons** means constraints. **Alog** means search algorithm.
S means single column index. **M** means multi-column index.
Welborn's work only focuses on single table.
DRL-index is not implemented yet.

Prior Work

Category	Work	Cost	Index type	IIA	Alog	Cons
Non-Learning method	AutoAdmin [VLD'97]	Estimated cost	S/M	✗	Greedy	index number
	ILP [ICDE'07]	Estimated cost	S/M	✗	ILP	storage
	ISRM [ICDE 19]	Estimated cost	S/M	✓	Greedy	storage
Learning-based method	AI Meet AI [SIGMOD'19]	Learning-model Estimated cost	S/M	Not sure	Greedy	index number
	Welborn et al [arxiv'19]	Not mention	S/M	✓	DQN	no
	DRL-Index [ICDEW'20]	Estimated cost	S	✓	DQN	Not mention

IIA means index interaction. **Cons** means constraints. **Alog** means search algorithm.
S means single column index. **M** means multi-column index.
Welborn's work only focuses on single table.
DRL-index is not implemented yet.

Prior Work

Category	Work	Cost	Index type	IIA	Alog	Cons
Non-Learning method	AutoAdmin [VLD'97]	Estimated cost	S/M	✗	Greedy	index number
	ILP [ICDE'07]	Estimated cost	S/M	✗	ILP	storage
	ISRM [ICDE 19]	Estimated cost	S/M	✓	Greedy	storage
Learning-based method	AI Meet AI [SIGMOD'19]	Learning-model Estimated cost	S/M	Not sure	Greedy	index number
	Welborn et al [arxiv'19]	Not mention	S/M	✓	DQN	no
	DRL-Index [ICDEW'20]	Estimated cost	S	✓	DQN	Not mention

Our Goal:

(1) Handle complex queries on multiple tables
(2) Recommend multi-column indexes
(3) Capture the index interaction

IIA means index interaction. **Cons** means constraints. **Alog** means search algorithm.
S means single column index. **M** means multi-column index.
Welborn's work only focuses on single table.
DRL-index is not implemented yet.

Our Method - Overview

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

workload

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(\boxed{W}, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

workload

Index configuration before starting the step t

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(\boxed{W}, \boxed{X_t}) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Index configuration before starting the step t

workload

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(\boxed{W}, \boxed{X_t}) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

workload

Index configuration before starting the step t

steps

$$\arg\max_{\boxed{\pi}} \sum_{t=0}^{\boxed{T}-1} (Cost(\boxed{W}, \boxed{X_t}) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \boxed{\pi}(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

workload

Index configuration before starting the step t

steps

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Index configuration before starting the step t

workload

steps

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Index configuration before starting the step t

workload

steps

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework

Workload Sample → Rules → Index Candidates →

Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

workload

Index configuration before starting the step t

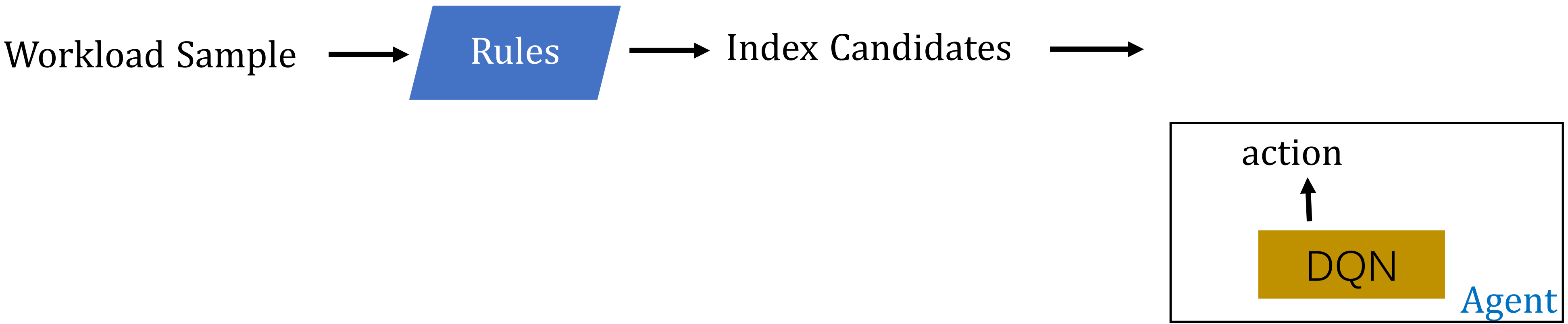
steps

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework



Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

steps

workload

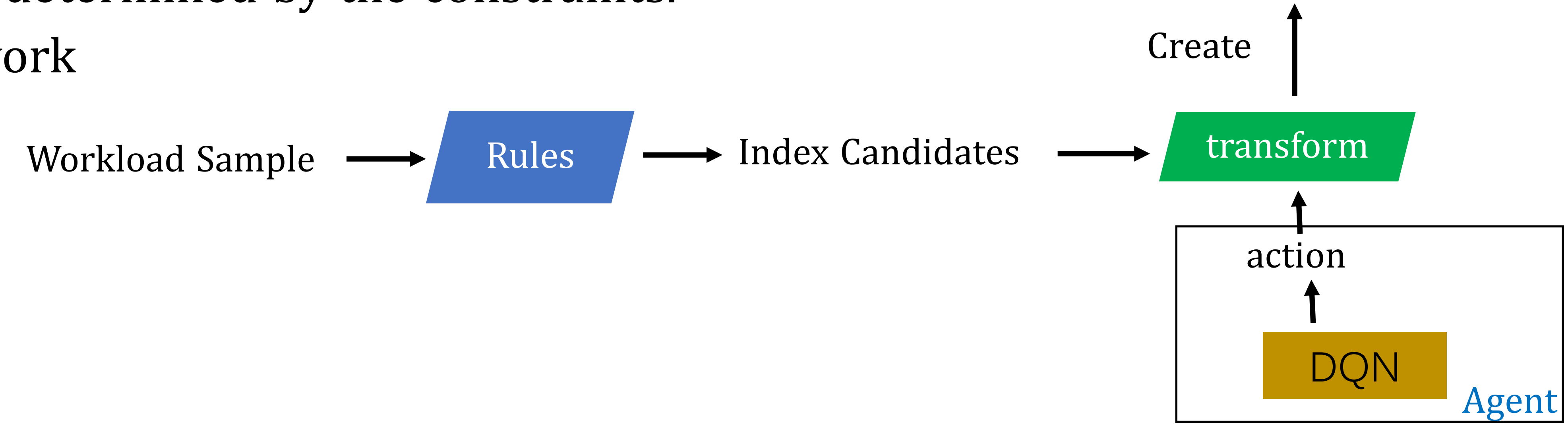
Index configuration before starting the step t

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework



Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Index configuration before starting the step t

workload

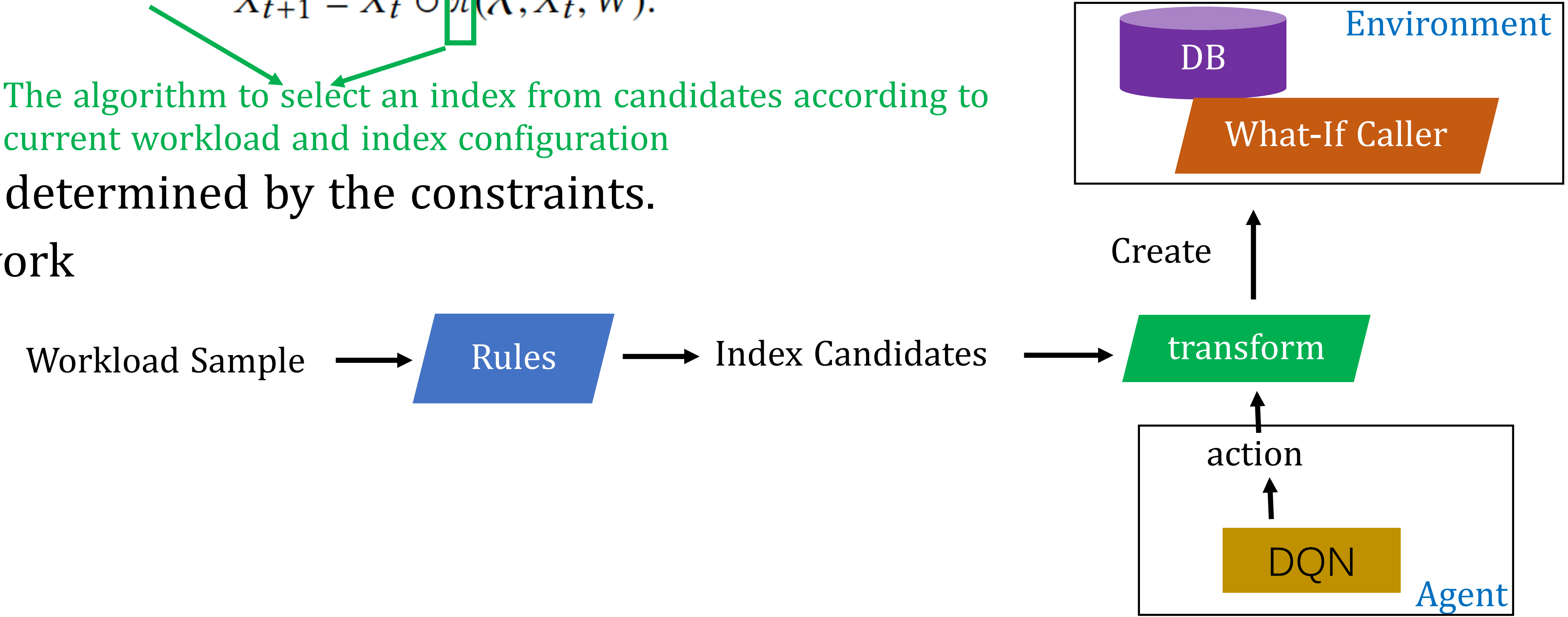
steps

$$\arg\max_{\pi} \sum_{t=0}^{T-1} (Cost(W, X_t) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \pi(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework



Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Index configuration before starting the step t

workload

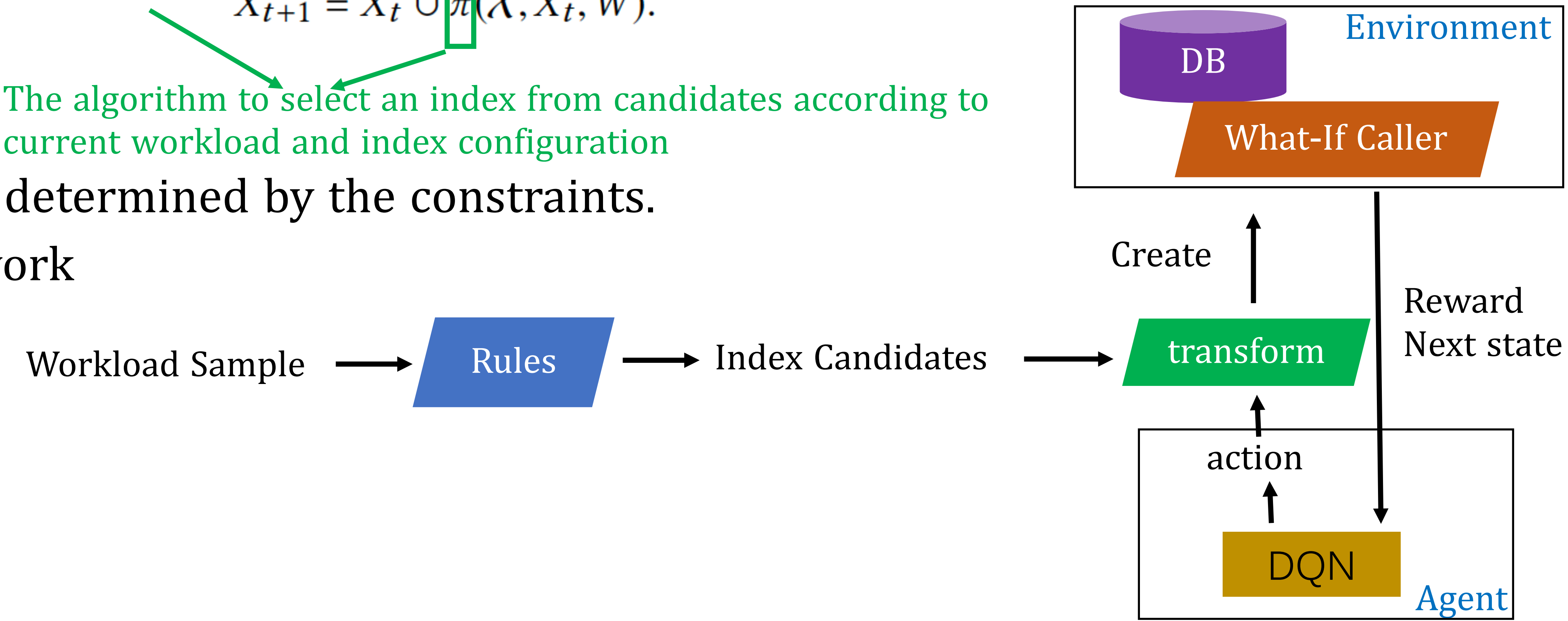
steps

$$\arg\max_{\boxed{\pi}} \sum_{t=0}^{\boxed{T}-1} (Cost(\boxed{W}, \boxed{X_t}) - Cost(W, X_{t+1}))$$

$$X_{t+1} = X_t \cup \boxed{\pi}(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework



Our Method - Overview

- Formulate Index Selection as a reinforcement learning problem
  - Maximize the Performance

Index configuration before starting the step t

workload

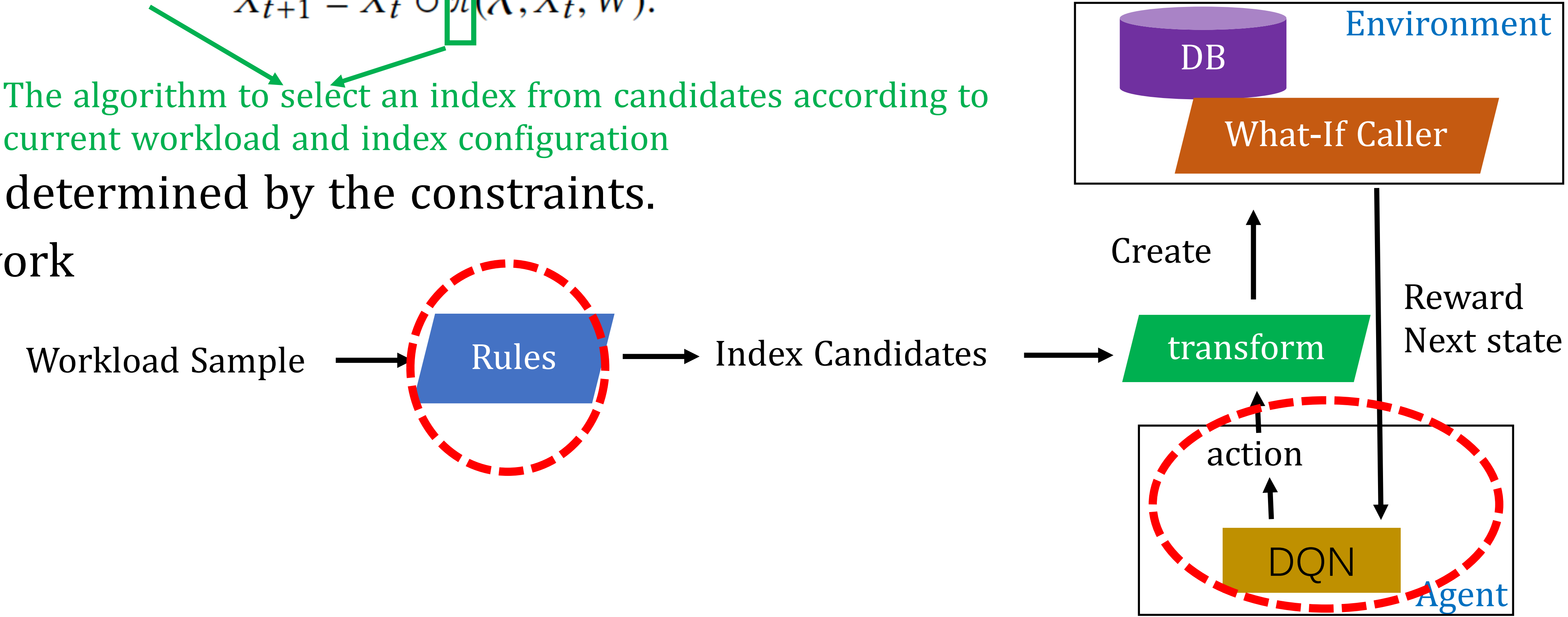
steps

$$\arg\max_{\boxed{\pi}} \sum_{t=0}^{\boxed{T}-1} \left(Cost(\boxed{W}, \boxed{X_t}) - Cost(W, X_{t+1})\right)$$

$$X_{t+1} = X_t \cup \boxed{\pi}(\mathcal{X}, X_t, W).$$

The algorithm to select an index from candidates according to current workload and index configuration

  - T is determined by the constraints.
- Framework



Our Method - Rules

Our Method - Rules

`J`: attributes that appear in JOIN conditions.
`EQ`: attributes that appear in EQUAL conditions.
`RANGE`: attributes that appear in RANGE conditions.
`O`: attributes that appear in GROUP BY, ORDER BY clauses.
`USED`: attributes that appear in this query.

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

```
SELECT t1.a7 FROM t1, t2
WHERE t1.a1 = t2.b1 AND t1.a2 = t2.b2
AND t1.a3 = 4 AND t2.b3 < 10
ORDER BY t1.a5, t1.a6
```

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

```
SELECT t1.a7 FROM t1, t2
WHERE t1.a1 = t2.b1 AND t1.a2 = t2.b2
AND t1.a3 = 4 AND t2.b3 < 10
ORDER BY t1.a5, t1.a6
```

↓

J: t1.a2, t1.a3, t2.b2, t2.b3
EQ: t1.a3
RANGE: t2.b3
O: t1.a5, t1.a6
USED: t1.(a1-a7), t2.(b1-b3)

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

```
SELECT t1.a7 FROM t1, t2
WHERE t1.a1 = t2.b1 AND t1.a2 = t2.b2
AND t1.a3 = 4 AND t2.b3 < 10
ORDER BY t1.a5, t1.a6
```

J: t1.a2, t1.a3, t2.b2, t2.b3
EQ: t1.a3
RANGE: t2.b3
O: t1.a5, t1.a6
USED: t1.(a1-a7), t2.(b1-b3)

Rule 1

t1.a1, t1.a2, t1.a3, t2.b1, t2.b2, t2.b3

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

```
SELECT t1.a7 FROM t1, t2
WHERE t1.a1 = t2.b1 AND t1.a2 = t2.b2
AND t1.a3 = 4 AND t2.b3 < 10
ORDER BY t1.a5, t1.a6
```

J: t1.a2, t1.a3, t2.b2, t2.b3
EQ: t1.a3
RANGE: t2.b3
O: t1.a5, t1.a6
USED: t1.(a1-a7), t2.(b1-b3)

Rule 1 → t1.a1, t1.a2, t1.a3, t2.b1, t2.b2, t2.b3

Rule 2 → (t1.a5, t1.a6)

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

SELECT t1.a7 FROM t1, t2
WHERE t1.a1 = t2.b1 AND t1.a2 = t2.b2
AND t1.a3 = 4 AND t2.b3 < 10
ORDER BY t1.a5, t1.a6

J: t1.a2, t1.a3, t2.b2, t2.b3
EQ: t1.a3
RANGE: t2.b3
O: t1.a5, t1.a6
USED: t1.(a1-a7), t2.(b1-b3)

Rule 1 → t1.a1, t1.a2, t1.a3, t2.b1, t2.b2, t2.b3

Rule 2 → (t1.a5, t1.a6)

Rule 3 → (t1.a1, t1.a2), (t2.b1, t2.b2), (t1.a2, t1.a1), (t2.b2, t2.b1)

Our Method - Rules

J: attributes that appear in JOIN conditions.
EQ: attributes that appear in EQUAL conditions.
RANGE: attributes that appear in RANGE conditions.
O: attributes that appear in GROUP BY, ORDER BY clauses.
USED: attributes that appear in this query.

Rule 1: Construct all single-attribute indexes by using the attributes in J, EQ, RANGE.
Rule 2: When the attributes in O come from the same table, generate the index by using all attributes in O.
Rule 3: If table *a* joins table *b* with multiple attributes, construct indexes by using all join attributes.

SELECT t1.a7 FROM t1, t2
WHERE t1.a1 = t2.b1 AND t1.a2 = t2.b2
AND t1.a3 = 4 AND t2.b3 < 10
ORDER BY t1.a5, t1.a6

J: t1.a2, t1.a3, t2.b2, t2.b3
EQ: t1.a3
RANGE: t2.b3
O: t1.a5, t1.a6
USED: t1.(a1-a7), t2.(b1-b3)

Rule 1: t1.a1, t1.a2, t1.a3, t2.b1, t2.b2, t2.b3

Rule 2: (t1.a5, t1.a6)

Rule 3: (t1.a1, t1.a2), (t2.b1, t2.b2), (t1.a2, t1.a1), (t2.b2, t2.b1)

Rule 4 & 5

Our Method - Model

- Key concepts in reinforcement learning model
  - The State records the information about current built indexes.
  - The Action in our model is choosing an index to build.
  - The Reward is defined:

Our Method - Model

- Key concepts in reinforcement learning model
  - The State records the information about current built indexes.
  - The Action in our model is choosing an index to build.
  - The Reward is defined:

$$r_t = \frac{Cost(W, X_{t-1}) - Cost(W, X_t))}{Cost(W, X_0)}$$

Our Method - Model

- Key concepts in reinforcement learning model
  - The State records the information about current built indexes.
  - The Action in our model is choosing an index to build.
  - The Reward is defined:

$$r_t = \frac{Cost(W, X_{t-1}) - Cost(W, X_t))}{Cost(W, X_0)}$$

- Why we choose DQN model?
  - The action space is discrete, which is the same with Q-Learning and DQN
  - Q-Learning is only effective for small state space. However the state space in ISP is quite large.
  - DDPG is the algorithm for learning continuous actions.

Experiments

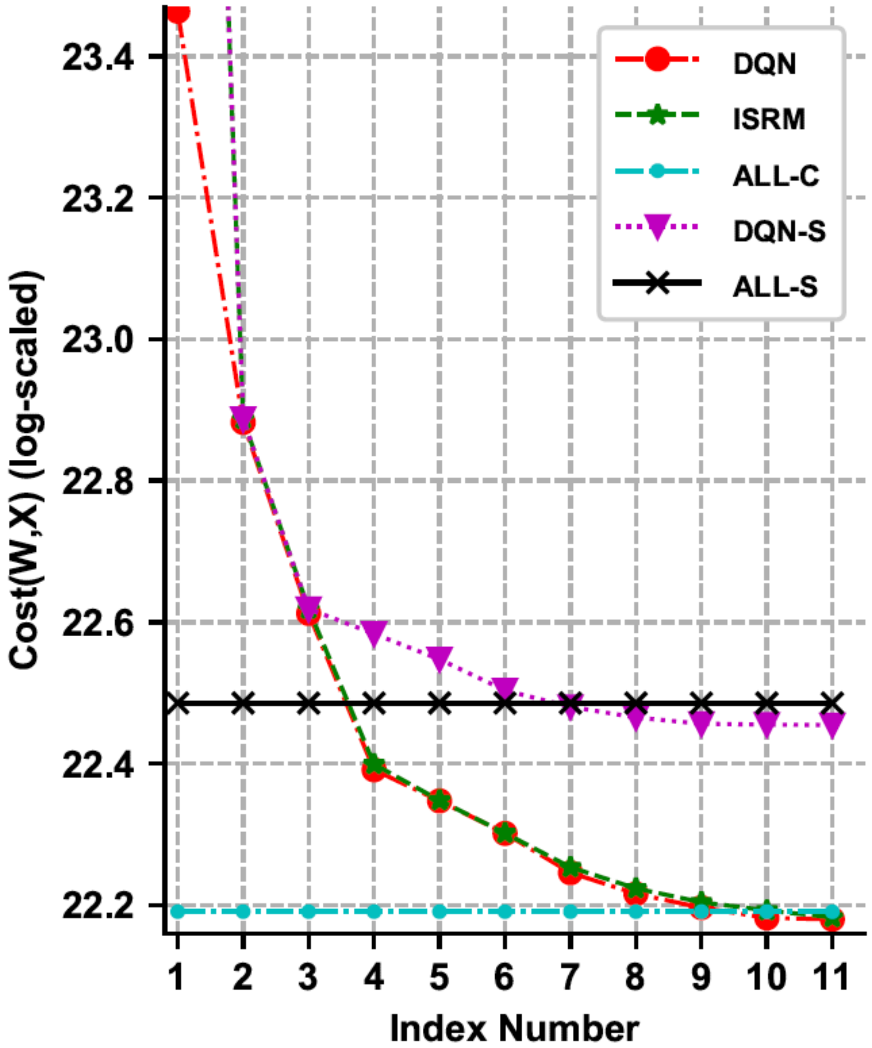
Question:

How well our method is compared with the current state-of-art method?

- Dataset: TPC-H with SF = 1
- Workload:
  - W^o (generated by the TPC-H query generator with 14 templates)
  - W^m (50 templates, queries on LINEITEM, multiple indexes)
- Evaluation Metric:
  - Estimated cost from optimizer
- Compared Methods:
  - ISRM [ICDE'19]

Experiments

- Index Selection on W^o for all tables



(1) W^o cannot get the best performance if only recommending single-attribute indexes by comparing ALL-S and ALL-C.

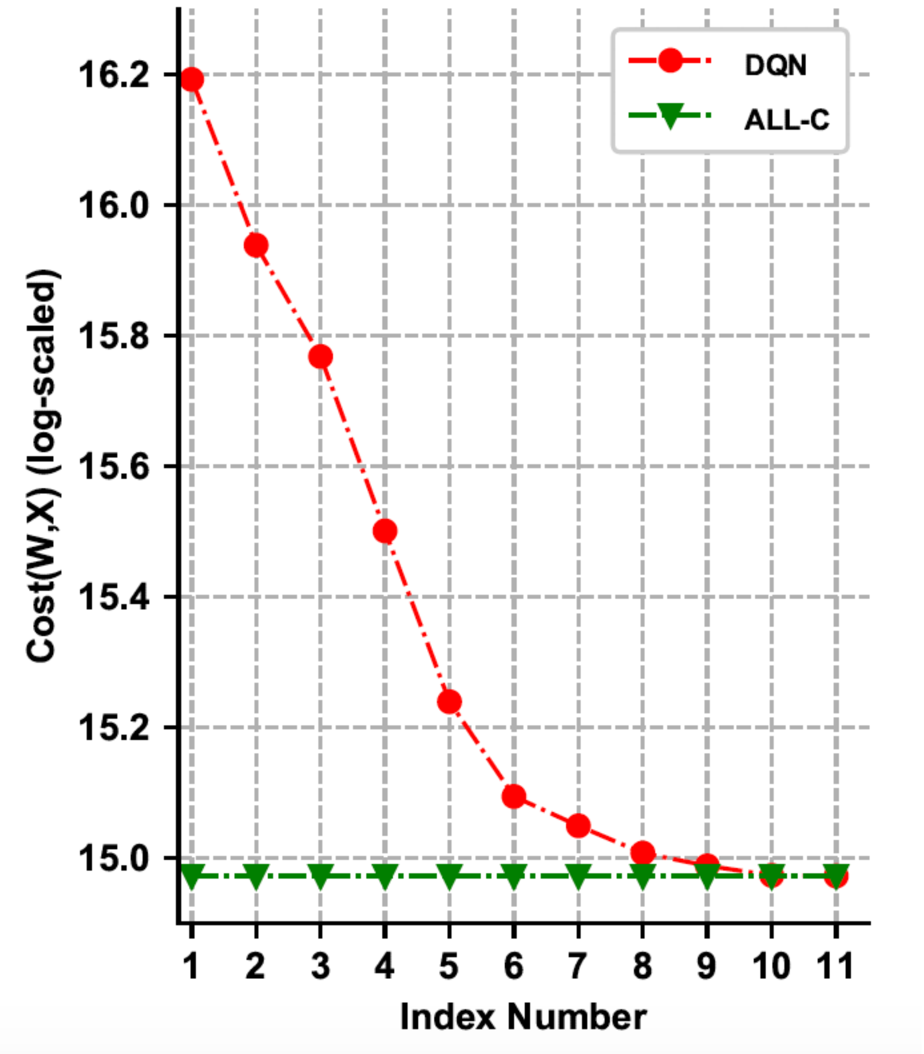
(2) When index number equals 1, the cost of W^o under DQN is much lower than DQN-S and ISMR.

(3) DQN-S and DQN get the optimal performance when index number is 7 and 10 separately. Even the costs of W^o under DQN-S and DQN can be lower than the optimal values.

(4) DQN is competitive to ISMR.

Experiments

- Index Selection on W^m



ISRM is sensitive to the order of attributes added in the algorithm.

Thank You
Q&A