An Index Advisor Using Deep Reinforcement Learning

Hai Lan¹, Zhifeng Bao¹, Yuwei Peng²

¹RMIT University
²Wuhan 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.

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

(1) An index on a  X
(2) An index on b  X
(3) An index on a and an index on b  ✓
```
Prior Work
## Prior Work

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

**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

<table>
<thead>
<tr>
<th>Category</th>
<th>Work</th>
<th>Cost</th>
<th>Index type</th>
<th>IIA</th>
<th>Alog</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Learning method</td>
<td>AutoAdmin [VLD'97]</td>
<td>Estimated cost</td>
<td>S/M</td>
<td>X</td>
<td></td>
<td>Greedy</td>
</tr>
<tr>
<td></td>
<td>ILP [ICDE'07]</td>
<td>Estimated cost</td>
<td>S/M</td>
<td>X</td>
<td></td>
<td>ILP</td>
</tr>
<tr>
<td></td>
<td>ISRM [ICDE 19]</td>
<td>Estimated cost</td>
<td>S/M</td>
<td>✓</td>
<td></td>
<td>Greedy</td>
</tr>
<tr>
<td>Learning-based method</td>
<td>AI Meet AI [SIGMOD'19]</td>
<td>Learning-model</td>
<td>S/M</td>
<td>Not sure</td>
<td>Greedy</td>
<td>index number</td>
</tr>
<tr>
<td></td>
<td>Welborn et al [arxiv'19]</td>
<td>Not mention</td>
<td>S/M</td>
<td>✓</td>
<td></td>
<td>DQN</td>
</tr>
<tr>
<td></td>
<td>DRL-Index [ICDEW'20]</td>
<td>Estimated cost</td>
<td>S</td>
<td>✓</td>
<td></td>
<td>DQN</td>
</tr>
</tbody>
</table>

## Prior Work

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

**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

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

\[
X_{t+1} = X_t \cup \pi(X, X_t, W).
\]
Our Method - Overview

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

\[
\text{workload} \quad \arg \max_{\pi} \sum_{t=0}^{T-1} (\text{Cost}(W, X_t) - \text{Cost}(W, X_{t+1}))
\]

\[
X_{t+1} = X_t \cup \pi(X, X_t, W).
\]
Our Method - Overview

• Formulate Index Selection as a **reinforcement learning problem**
• Maximize the Performance

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

\[X_{t+1} = X_t \cup \pi(X, X_t, W).\]
Our Method - Overview

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

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

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

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

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

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

- \( T \) is determined by the constraints.
Our Method - Overview

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

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

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

• 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

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

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

- The algorithm to select an index from candidates according to current workload and index configuration
- \( T \) is determined by the constraints.
- Framework

Workload Sample \(\rightarrow\) Rules \(\rightarrow\) Index Candidates
Our Method - Overview

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

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

\[ \underset{\pi}{\arg \max} \sum_{t=0}^{T-1} (\text{Cost}(W, X_t) - \text{Cost}(W, X_{t+1})) \]

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

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 → transform

Create

action

DQN Agent
Our Method - Overview

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

\[
\arg\max_{\pi} \sum_{t=0}^{T-1} (\text{Cost}(W_{t}, X_{t}) - \text{Cost}(W, X_{t+1})).
\]

- 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

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

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

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 → transform → DQN → Agent

Create → Reward Next state → What-If Caller

Environment → DB
Our Method - Overview

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

\[
\text{arg max } \sum_{t=0}^{T-1} (\text{Cost}(W, X_t) - \text{Cost}(W, X_{t+1}))
\]

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

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.

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

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.

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.

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

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.

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

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.

Rule 1: t1.a1, t1.a2, t1.a3, t2.b1, t2.b2, t2.b3
Rule 2: (t1.a5, t1.a6)

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.

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: 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.

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

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.

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 - 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:

\[
r_t = \frac{\text{Cost}(W, X_{t-1}) - \text{Cost}(W, X_t))}{\text{Cost}(W, X_0)}
\]
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:

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

• 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 $W^0$ for all tables

(1) $W^0$ 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 $W^0$ 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 $W^0$ under DQN-S and DQN can be lower than the optimal values.

(4) DQN is competitive to ISMR.
ISRM is sensitive to the order of attributes added in the algorithm.
Thank You

Q&A