



Match Plan Generation in Web Search with Parameterized Action Reinforcement Learning

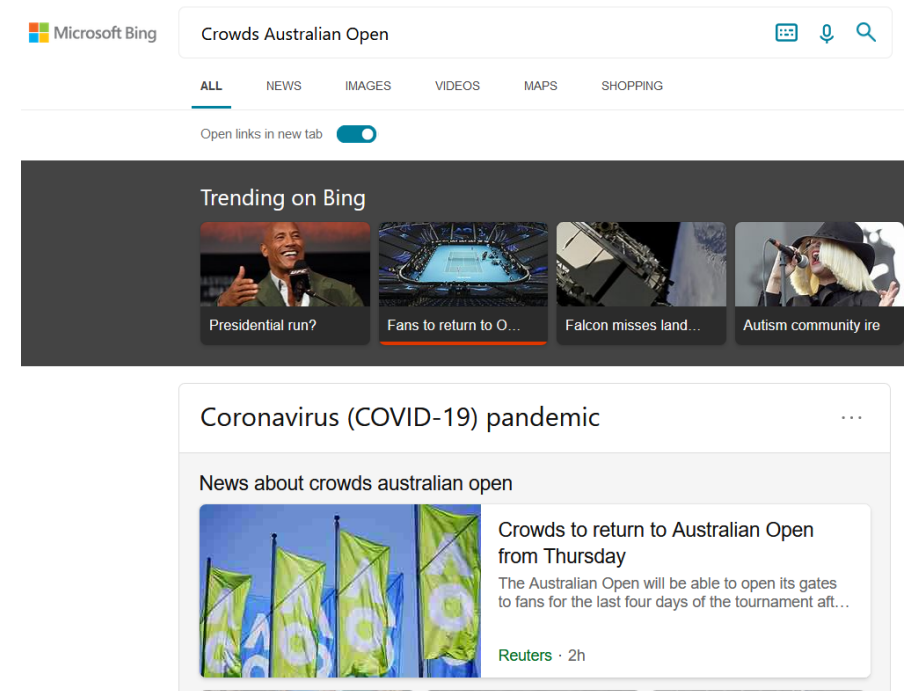
Ziyan Luo*, Linfeng Zhao*, Wei Cheng*, Sihao Chen, Qi Chen, Hui Xue,
Haidong Wang, Chuanjie Liu, Mao Yang, Lintao Zhang

discat@foxmail.com, zhao.linf@northeastern.edu, weicheng5993@foxmail.com, sihao@berkeley.edu
{cheqi,xuehui,haidwa,chuanli,maoyang,lintaoz}@microsoft.com

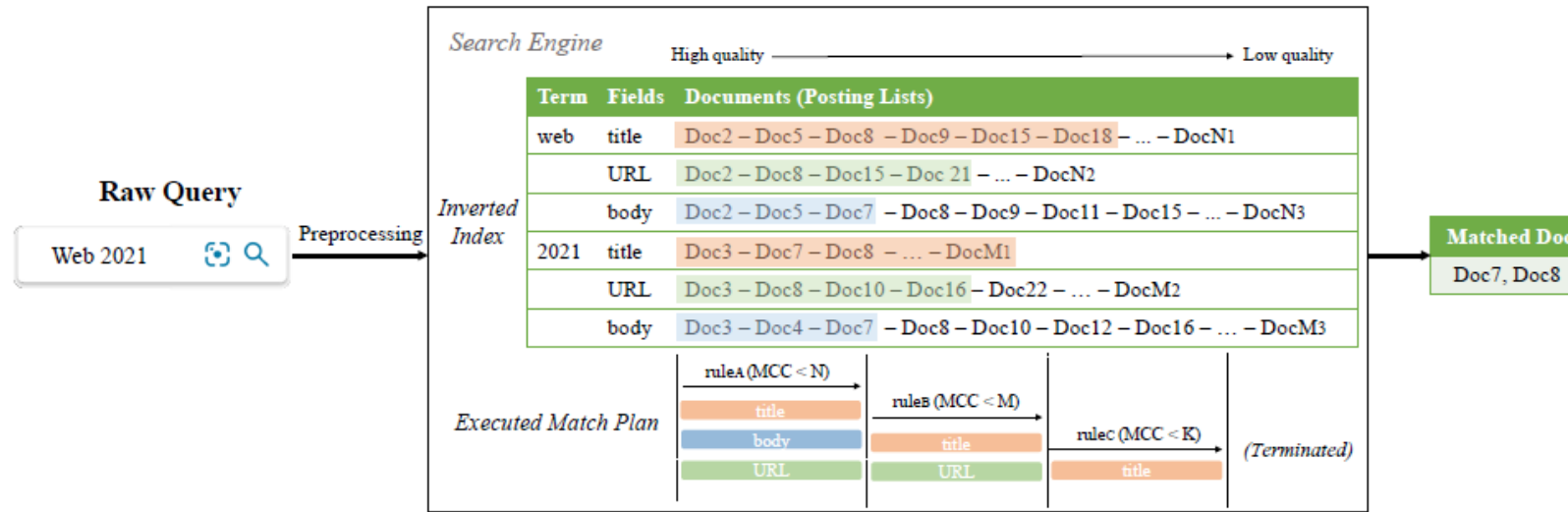
Introduction

- Match plan generation is the key technology for large scale search engines
- Aims
 - **1. Good result quality (relevance)**
 - **2. Short query response time**
- Search engines use match plans to help retrieve relevant documents from billions of web pages

Microsoft Bing



Match Plan Generation Process



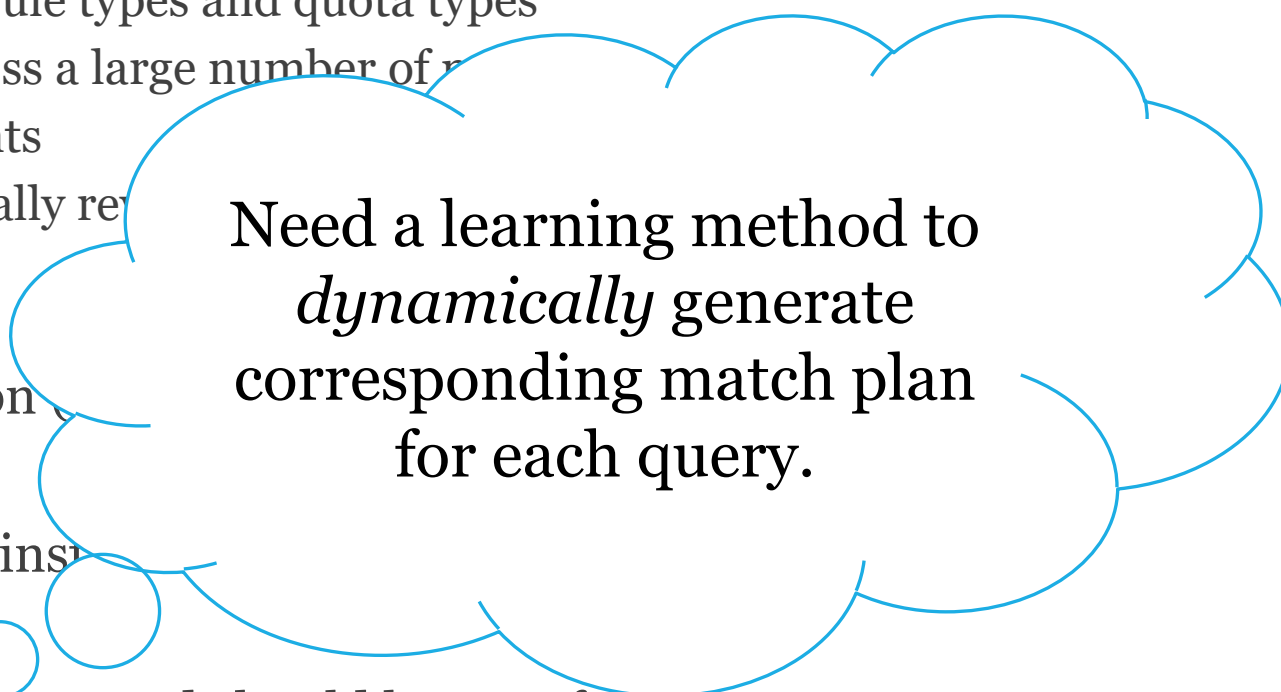
- After preprocessing, multiple posting lists are retrieved.
- The search engine scans them by executing a *match plan* which is composed of a sequence of *match rules*.
- A *match rule* defines *how* the search engine matches documents over a period.
- It is made up of a **discrete match rule type** (e.g. $rule_A$) and several **continuous stopping quotas** (e.g. $MCC < N$).
- Different match rules have different execution costs.

Match Plan Plays critical role in Web Search

- Help to retrieve top candidates in milliseconds.
- Decide the resource allocation for a query.
- Help to make the trade-off between *relevance* and *efficiency*.
- It's a secret for search companies.
 - No publication, no open source.
 - Open toolkits (e.g. Lucene, Elasticsearch) do not have similar strategy.

Why generating match plans is hard?

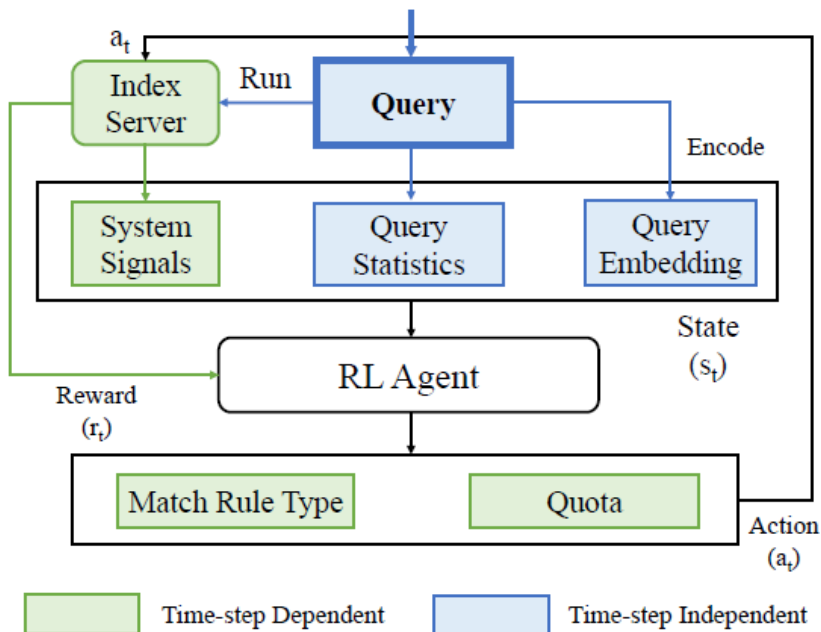
- The complexity of the system environment
 - Increasing number of match rule types and quota types
 - Diverse data distribution across a large number of machines
 - Frequent updates of documents
 - Static design cannot dynamically reconfigure



Need a learning method to
dynamically generate
corresponding match plan
for each query.

- Multiple objects optimization
- Sequence decision making (instance)
- Apply in thousands of machines and should be very fast

Problem Formulation



A POMDP, a tuple of $(S, A, P, R, \Omega, O, \gamma)$

State

- Intermediate System Signals
- Query Embeddings, Statistics

Action $\mathcal{A} = \{(k, x) | k \in \mathcal{A}_d, x \in \mathcal{X}\} = \mathcal{A}_d \times \mathcal{X}$,

- **Discrete**: m types of predefined match rules + *Stop*
- **Continuous (shared)**: n dims of Quotas

Reward a scalar function weighted by:

- Performance: “**Relevance Scores**” (**RS**) of top k matched documents (From Bing’s server)
- Latency: “**Index Block Accesses**” (**IBA**) of the match plan in the system

$$r_t = (\lambda_1 RS_t - \lambda_2 IBA_t) - (\lambda_1 RS_{t-1} - \lambda_2 IBA_{t-1}),$$

Environment Bing’s index server (wrapped)

Could We Use Existed RL Algorithm?

Algorithm	DQN	TD3	SAC	PA-DDPG	What we expect
Discreate action	✓	×	✓	✓	✓
Continuous action	×	✓	✓	✓	✓
Discreate & Continuous action	×	×	×	✓	✓
Stability	×	×	✓	×	✓
Performance (<i>better than production</i>)	×	✓	✓	×	✓

- Complex action space
 - *Complex action: combine discrete and continuous spaces*
 - *Huge action space $17,249,876,309 * 10^{105}$*
- Instability in training
 - *Due to the lack of exploration*
- Sampling deviation in traditional prioritized replay buffer
 - *Experiences whose rewards are in certain ranges are more likely to be sampled, making the agent behave poorly in some state subspaces*
 - *Cause poor performance of learning the value function for some queries*

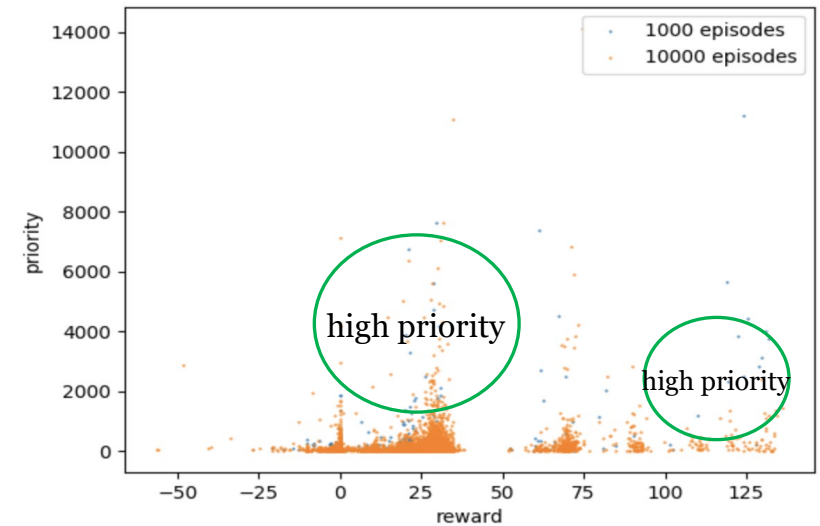
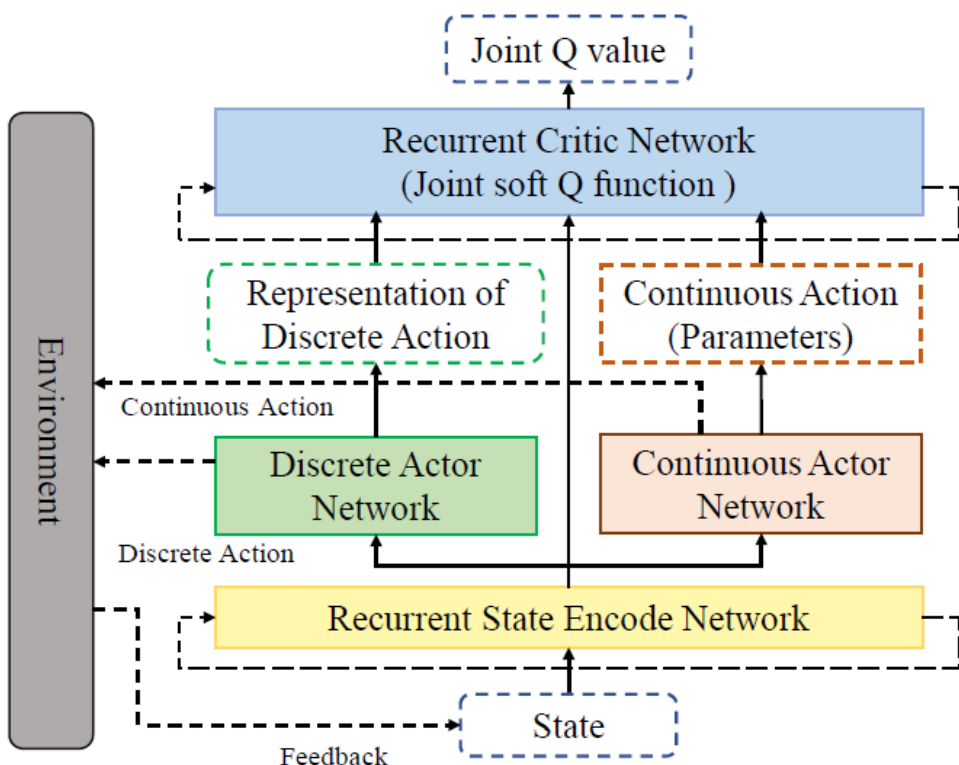


Figure 4. The deviation in original PER.

Parameterized Action Soft Actor-Critic



Challenges

- Parameterized (discrete-continuous hybrid) action space
- Complex environment, large state/action space
- Sparse reward, partial observability

PASAC:

- 1. Optimize a **stochastic policy** of the *complete* action: discrete match rules (Categorical dist.) and continuous quotas (Gaussian dist.), meanwhile maximize both entropies

$$J_{\pi}(\phi) = \mathbb{E}_{s_t \sim D} \left[\mathbb{E}_{k_t \sim \pi_{\phi}} \left[\alpha_d \log \left(\pi_{\phi}(k_t | s_t) \right) - Q_{\theta}(s_t, k_t, x_t) \right] \right] \quad (3)$$

$$J_{\pi}(\psi) = \mathbb{E}_{s_t \sim D} \left[\mathbb{E}_{x_t \sim \pi_{\psi}} \left[\alpha_c \log \left(\pi_{\psi}(x_t | s_t) \right) - Q_{\theta}(s_t, k_t, x_t) \right] \right] \quad (4)$$

- 2. Soft Q network: estimate a joint soft Q-value function for the *complete* action

$$J_Q(\theta) = \mathbb{E}_{(s_t, x_t, k_t) \sim D} \left[\frac{1}{2} \left(Q_{\theta}^t(s_t, x_t, k_t) - y_t \right)^2 \right]$$

Parameterized Action Soft Actor-Critic

Algorithm 1 Parameterized Action Soft Actor-Critic

input: Initial parameters $\theta_1, \theta_2, \phi, \psi$
 \diamond Initialize target net weights $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$
 \diamond Initialize an replay buffer $\mathcal{D} \leftarrow \emptyset$
 for each episode **do**
 for each environment step **do**
 \diamond Sample discrete action from the policy $k_t \sim \pi_\phi(k_t|s_t)$
 \diamond Sample parameter from the policy $x_t \sim \pi_\psi(x_t|s_t)$
 \diamond Store the transition $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, f(k_t), x_t, r_t, s_{t+1})$
 end for
 for each gradient step **do**
 \diamond Sample a mini-batch from replay buffer \mathcal{D}
 \diamond Update the joint soft Q-function parameters
 $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in [1, 2]$
 \diamond Update discrete policy weights
 $\phi \leftarrow \phi - \lambda_{\pi_\phi} \nabla_{\phi} J_\pi(\phi)$
 \diamond Update continuous policy weights
 $\psi \leftarrow \psi - \lambda_{\pi_\psi} \nabla_{\psi} J_\pi(\psi)$
 \diamond Adjust temperature of discrete policy's entropy
 $\alpha_d \leftarrow \alpha_d - \lambda_{\alpha_d} \nabla_{\alpha_d} J(\alpha_d)$
 \diamond Adjust temperature of continuous policy's entropy
 $\alpha_c \leftarrow \alpha_c - \lambda_{\alpha_c} \nabla_{\alpha_c} J(\alpha_c)$
 \diamond Update target network weights
 $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau) \bar{\theta}_i$ for $i \in [1, 2]$
 end for
 end for
output: Optimized parameters $\theta_1, \theta_2, \phi, \psi$

Implementation Details:

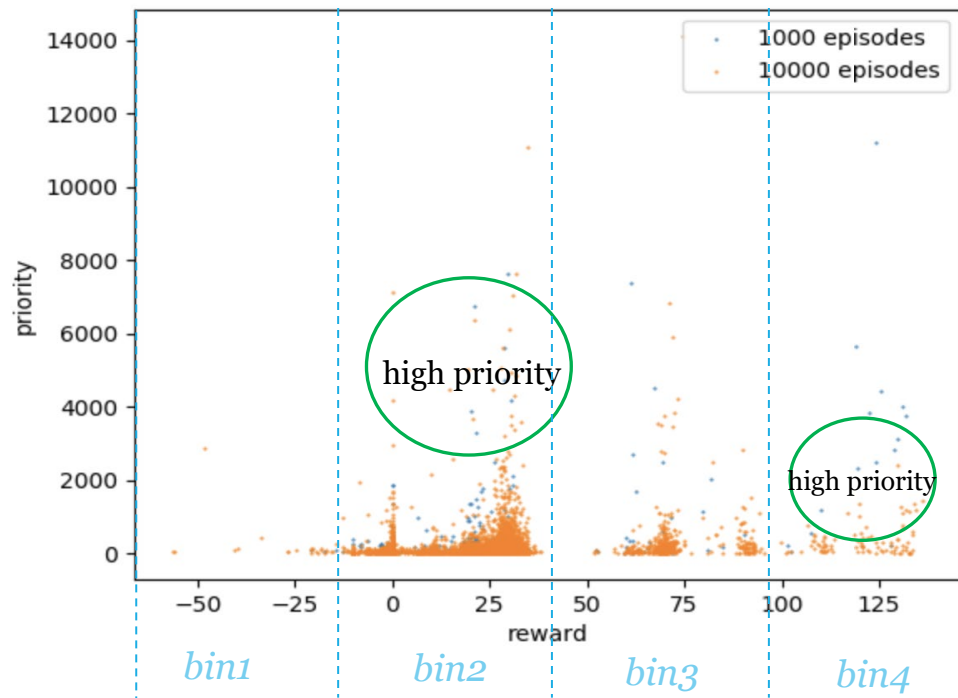
- **Exploration:** *double alpha tuning* to control the different exploration rate at the discrete and continuous action spaces

$$J(\alpha_d) = \mathbb{E}_{k_t \sim \pi_\phi^t} \left[-\alpha_d \left(\log \left(\pi_\phi(k_t|s_t) \right) + \bar{\mathcal{H}}_d \right) \right]$$

$$J(\alpha_c) = \mathbb{E}_{x_t \sim \pi_\psi^t} \left[-\alpha_c \left(\log \left(\pi_\psi(x_t|s_t) \right) + \bar{\mathcal{H}}_c \right) \right]$$

- **Recurrent state head:** dynamic LSTM to solve the Partially Observation problem

Stratified PER



We further proposed Stratified Prioritized Experience Replay (SPER) to address the “*skewed prioritizing*” issue:

- **skewed prioritizing:** experiences whose rewards are in certain ranges are more likely to be sampled, making the agent behave poorly in some state subspaces (some queries are inherently easy/hard to train)
- **buffer stratifying:** the replay buffer is divided into several bins (strata) according to reward range. The same number of samples are sampled from each bin by important sampling
- **priority with TD-error and *policy loss*:**
 - Transactions with larger improvement potential more likely to be sampled

$$p(s_t, a_t) = |\delta(s_t, a_t)| + \lambda \xi(s_t, a_t) + \epsilon_d,$$

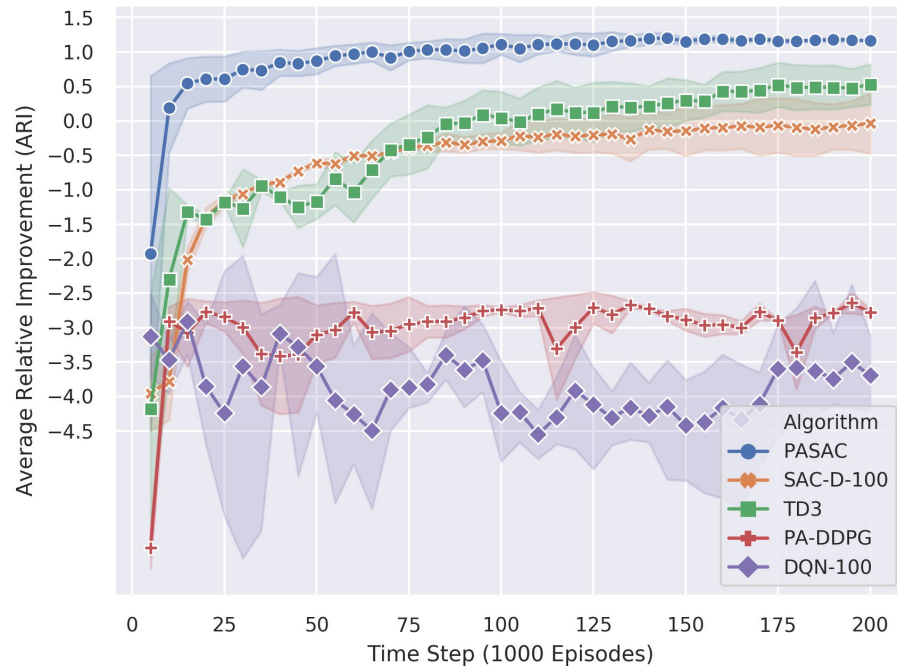
Experiments

- Q1. Does the proposed algorithm work better than the heuristic hand-crafted method tuned by engineers, or other RL algorithms?
- Q2. Is it more appropriate that we formulate the problem into a PARL problem, instead of discretizing the action space?
- Q3. How is the improvement of our method in real search scenes?
- Q4. How is the effect of applying SPER, and its components?
- Q5. Does the proposed agent work well on other PARL benchmarking baselines?

Experiments

Q1,2,3: Some comparative experiments with the same condition (3,000 test queries in total)

For different RL agents (Figure on ARI)



$$ARI = \frac{\sum_{i=1}^{|D|} (r_{agent}^i - r_{baseline}^i)}{|D|}$$

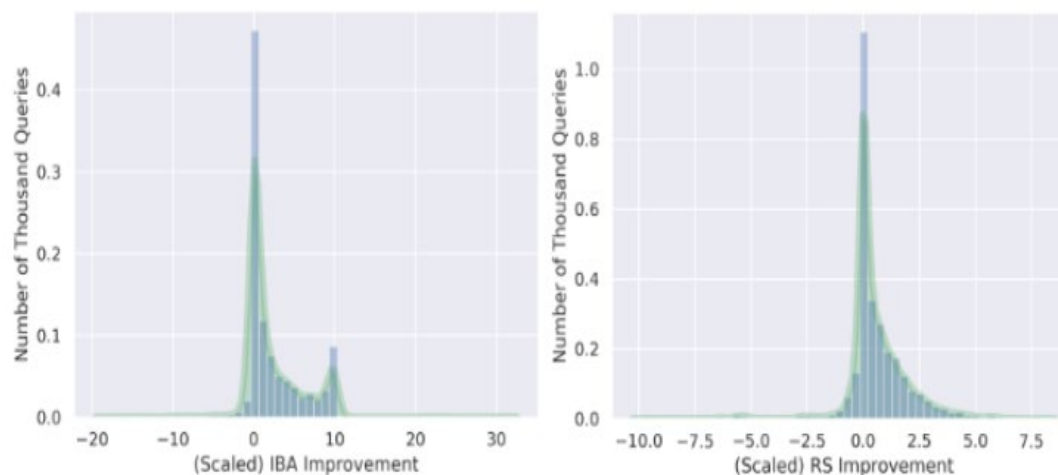
For different RL agents (Table)

	DQN-20	DQN-100	D-SAC-20	D-SAC-100	PA-DDPG	TD3	PASAC	PASAC+SPER
ARI	-1.500	-1.957	0.210	0.460	-2.492	0.8384	1.280	1.912
Better	26.70%	27.43%	40.47%	49.30%	39.97%	41.90%	50.53%	60.10%
Equal	3.70%	4.50%	10.10%	6.03%	3.17%	4.67%	6.07%	11.60%

PASAC agent apparently outperforms other SOTA agents in both stability and efficiency in our scene

Experiments

Performance Improvements for Production



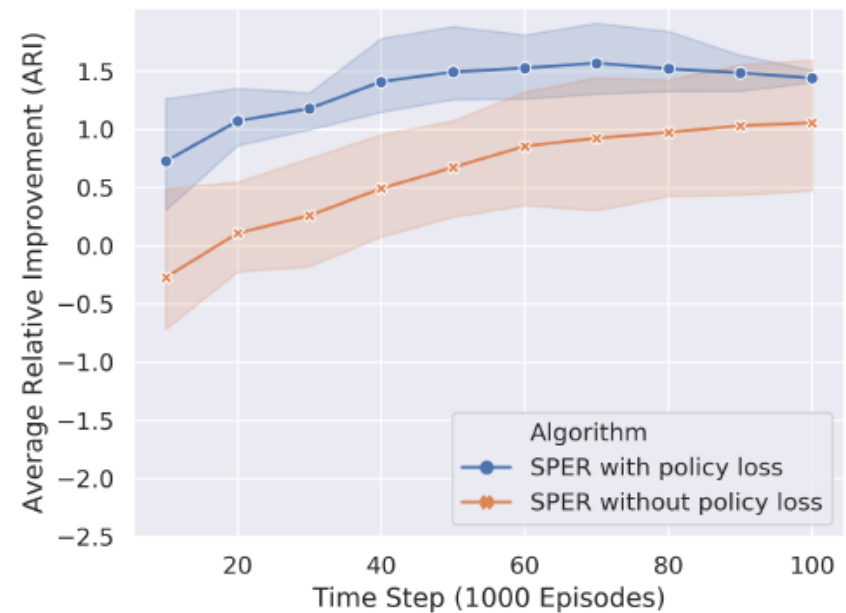
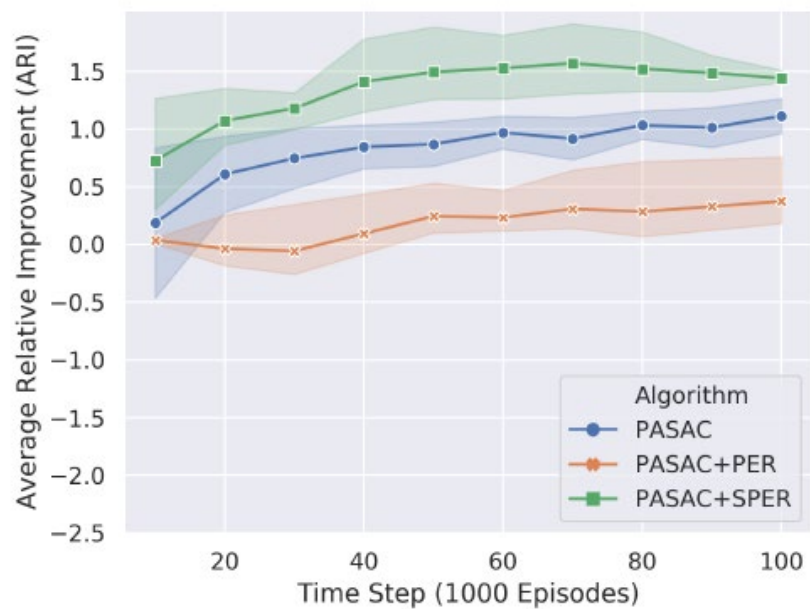
	IBA	RS	Latency	Latency+inference
Improvement	+75.77%	-1.47%	+28.14%	+8.97%

- **Significant reduction of index block accesses** with relevance on-par
 - *Manually defined match plans cannot flexibly control the quotas*
- Match plans generated by model are typically **shorter** than production
 - *Production rules are generalized to all queries in a category, leading to some redundancy rules for a single query*

Experiments

Ablation Study

Q4. How is the effect of applying SPER, and its components?



Experiments

Benchmark Games

Q5: Does the proposed agent work well on other PARL benchmarking baselines?

We evaluate our agent in a broader context

Table 3: Average evaluation results (the average of all training rewards and final evaluation reward (repeated 100 times)) on benchmarks Platform-v0 and Goal-v0 with PA-DDPG [8].

Average Eval Return	PASAC	PASAC+SPER	PA-DDPG[8]
Platform-v0	0.9723	0.9727	0.3113
Goal-v0	43.11	43.85	-6.208

- PASAC performs much better than PA-DDPG.
- Stratified sampling may better fit the environment with skewed prioritizing issue if PER is applied.

Summary

- Formulate the match plan generation task to the general PARL framework
 - Propose a novel algorithm, Parameterized Action Soft Actor-Critic
 - To address the *skewed prioritizing* issue of PER, Stratified Prioritized Experience Replay (SPER) is applied
- Experiment results show that our learned match plan significantly outperforms the production baseline in terms of resource-saving
- Future works include further optimize the model reference time, and inventing more delicate strategies in exploring the parameterized action space



THE **WEB**
CONFERENCE



Thank you
for your careful listening!