MS&E 233
Game Theory, Data Science and AI
Lecture 8

Vasilis Syrgkanis

Assistant Professor

Management Science and Engineering

(by courtesy) Computer Science and Electrical Engineering

Institute for Computational and Mathematical Engineering

Computational Game Theory for Complex Games

1

- Basics of game theory and zero-sum games (T)
- Basics of online learning theory (T)
- Solving zero-sum games via online learning (T)
- *HW1: implement simple algorithms to solve zero-sum games*
- Applications to ML and AI (T+A)
- *HW2: implement boosting as solving a zero-sum game*

2

- Basics of extensive-form games
- Solving extensive-form games via online learning (T)
- *HW3: implement agents to solve very simple variants of poker*

3

- General games, equilibria and online learning (T)
- **Online learning in general games**
- *HW4: implement no-regret algorithms that converge to correlated equilibria in general games*

Data Science for Auctions and Mechanisms

4

- Basics and applications of auction theory (T+A)
- Learning to bid in auctions via online learning (T)
- *HW5: implement bandit algorithms to bid in ad auctions*

5

- Optimal auctions and mechanisms (T)
- Simple vs optimal mechanisms (T)
- *HW6: calculate equilibria in simple auctions, implement simple and optimal auctions, analyze revenue empirically*

6

- Optimizing mechanisms from samples (T)
- Online optimization of auctions and mechanisms (T)
- *HW7: implement procedures to learn approximately optimal auctions from historical samples and in an online manner*

Further Topics

7

- Econometrics in games and auctions (T+A)
- A/B testing in markets (T+A)
- *HW8: implement procedure to estimate values from bids in an auction, empirically analyze inaccuracy of A/B tests in markets*

Guest Lectures

- Mechanism Design and LLMs, Song Zuo, Google Research
- A/B testing in auction markets, Okke Schrijvers, Central Applied Science, Meta

Recap: Regret vs Correlated Equilibrium

Distributions that satisfy this are called **Coarse Correlated Equilibria**

- No-regret property, implies

$$\forall s_i': \sum_{s} \pi^T(s)\left(u_i(s) - u_i(s_i', s_{-i})\right) \geq -\tilde{\epsilon}(T,\delta) \rightarrow 0$$

- Correlated equilibrium requires conditioning on recommendation

$$\forall s_i^*, s_i': \sum_{s: s_i = s_i^*} \pi^T(s)\left(u_i(s) - u_i(s_i', s_{-i})\right) \geq 0$$

At subset of periods when **played s_i^***

You don't regret **switching to s_i'**

Recap: Swaps and Correlated Equilibrium

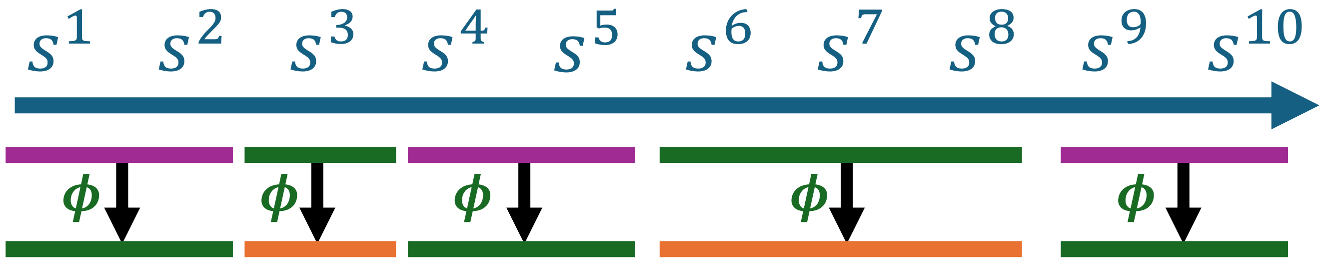
- Correlated equilibrium requires conditioning on recommendation

$$\forall s_i^*, s_i': \sum_{s: s_i = s_i^*} \pi^T(s)\left(u_i(s) - u_i\left(s_i', s_{-i}\right)\right) \geq 0$$

- Equivalently: for any **swap** function ϕ that maps original actions s_i to deviating actions s_i' (potentially different for each original s_i)

$$\sum_s \pi^T(s)\big(u_i(s) - u_i(\phi(s_i), s_{-i})\big) \geq 0$$

At all periods



You don’t regret **swapping your original action based on the mapping ϕ**

Recap: No-Swap Regret!

- No-regret property requires

$$\frac{1}{T}\sum_{t=1}^{T} u_i\left(s^t\right) \geq \max_{s_i' \in S_i} \frac{1}{T}\sum_{t=1}^{T} u_i\left(s_i', s_{-i}^t\right) - \tilde{\epsilon}(T, \delta)$$

- No-swap regret property requires

$$\forall \phi: \frac{1}{T}\sum_{t=1}^{T} u_i\left(s^t\right) \geq \frac{1}{T}\sum_{t=1}^{T} u_i\left(\phi\left(s_i^t\right), s_{-i}^t\right) - \tilde{\epsilon}(T, \delta)$$

Theorem. If all players use no-swap regret algorithms, then the empirical joint distribution converges to a Correlated Equilibrium

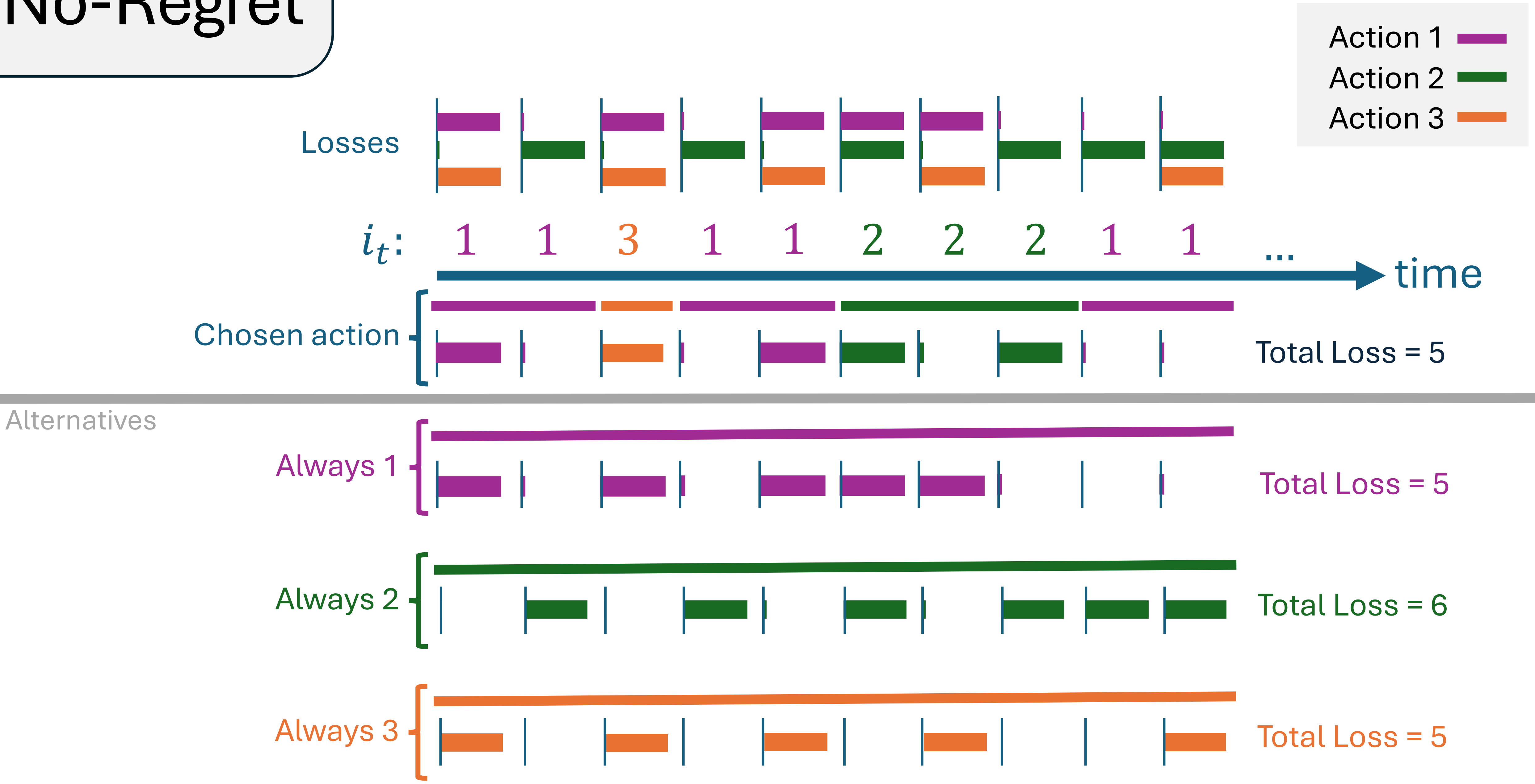
Can we construct algorithms with vanishing no-swap regret?

No Swap Regret vs No Regret

- At period t you choose action i_t from distribution x_t over n actions
- Observe vector $\ell_t = \left(\ell_t^1, \ldots, \ell_t^n\right)$ containing loss of each action
- You incur the loss of the action you chose $\ell_t^{i_t}$
- No-regret: for any action i, you do not regret always taking action i

$$\frac{1}{T}\sum_t \ell_t^{i_t} \le \frac{1}{T}\sum_t \ell_t^i + \tilde{\epsilon}(T, \delta), \qquad \text{w.p. } 1-\delta$$

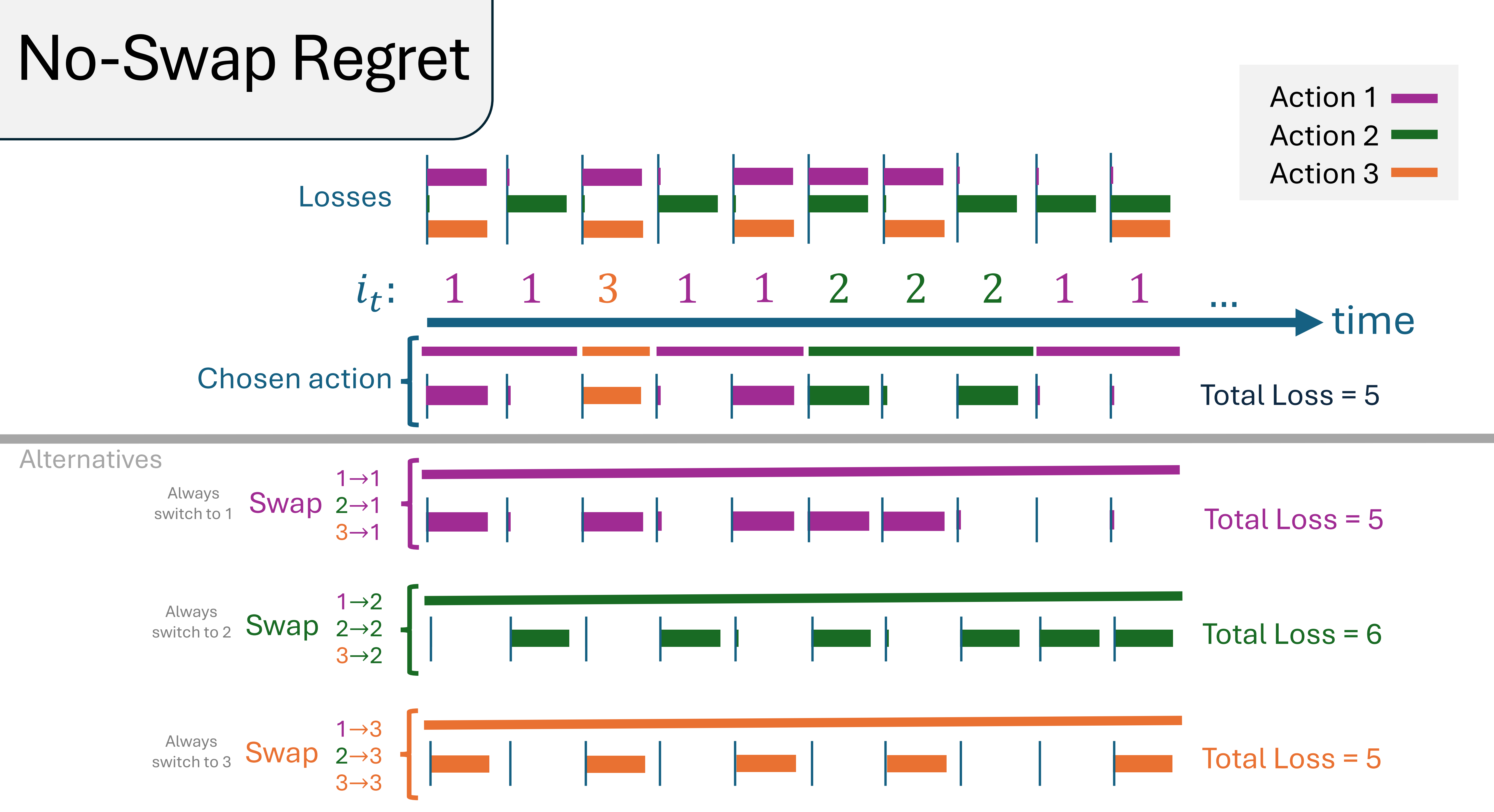
No-Regret


Action 1
Action 2
Action 3
Losses
i_t: 1 1 3 1 1 2 2 2 1 1 ...
time
Chosen action
Total Loss = 5
Alternatives
Always 1
Total Loss = 5
Always 2
Total Loss = 6
Always 3
Total Loss = 5

No Swap Regret vs No Regret

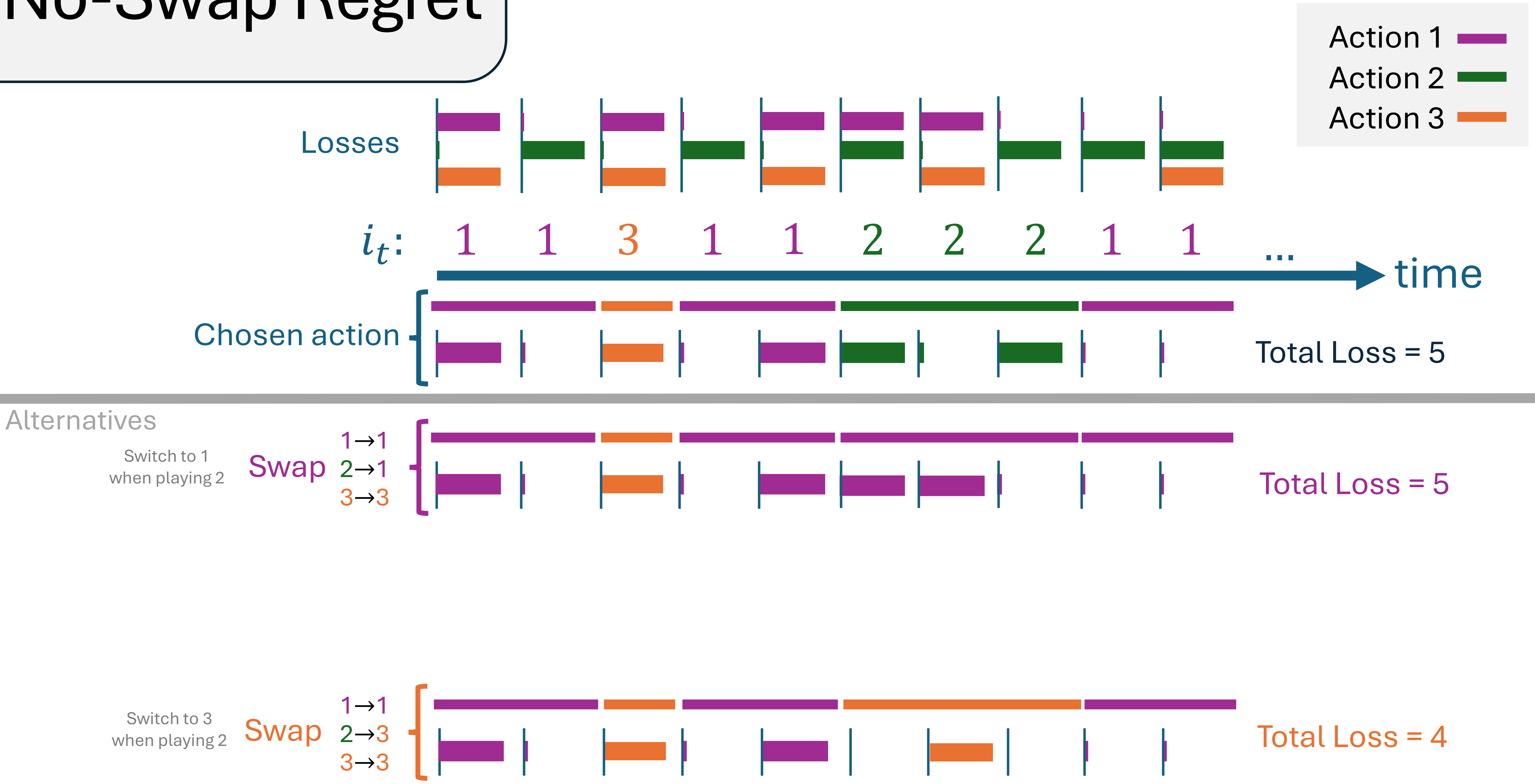
- At period t you choose action i_t from distribution x_t over n actions
- Observe vector $\ell_t = \left(\ell_t^1, \dots, \ell_t^n\right)$ containing loss of each action
- You incur the loss of the action you chose $\ell_t^{i_t}$
- No-swap regret: for any swap function ϕ mapping original actions i to alternatives $i' = \phi(i)$, you do not regret making that swap

$$\frac{1}{T}\sum_t \ell_t^{i_t} \leq \frac{1}{T}\sum_t \ell_t^{\phi(i_t)} + \tilde{\epsilon}(T, \delta), \qquad \text{w.p. } 1 - \delta$$



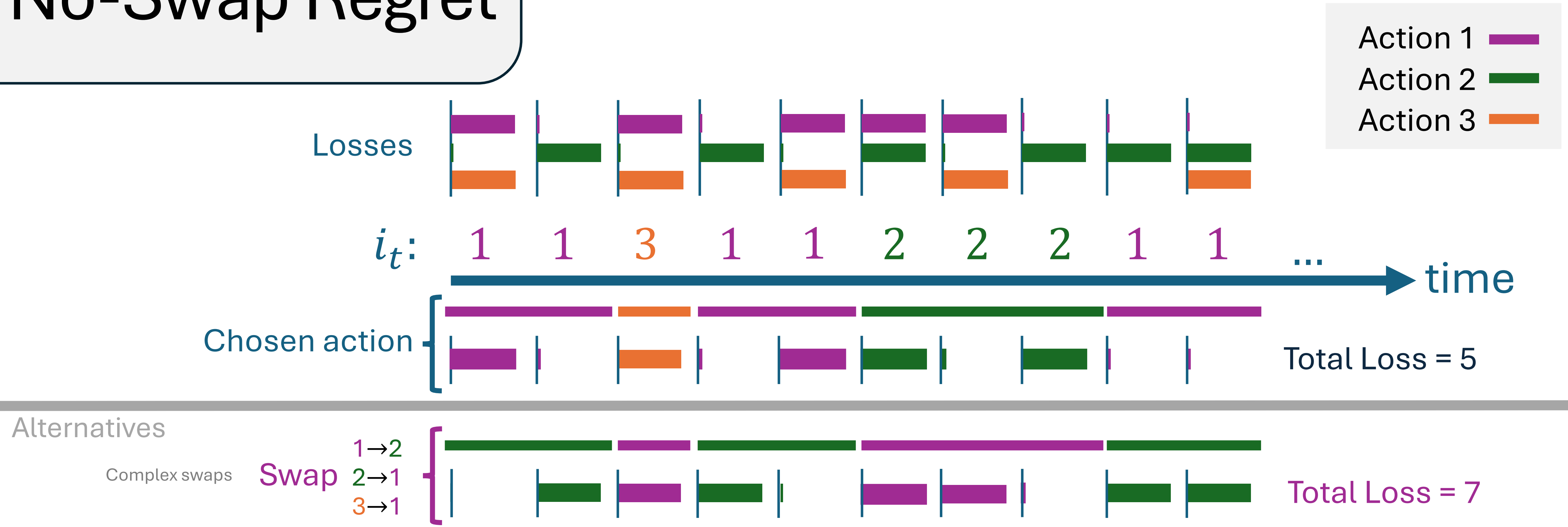
No-Swap Regret
Action 1
Action 2
Action 3
Losses
i_t: 1 1 3 1 1 2 2 2 1 1 ...
time
Chosen action
Total Loss = 5
Alternatives
Always switch to 1
Swap
1→1
2→1
3→1
Total Loss = 5
Always switch to 2
Swap
1→2
2→2
3→2
Total Loss = 6
Always switch to 3
Swap
1→3
2→3
3→3
Total Loss = 5

No-Swap Regret



No-Swap Regret

No-Swap Regret



Vanishing regret for complex swaps is implied by vanishing regret of simple swaps: switch to j' whenever you had played j and leave everything else as is

No Swap Regret vs No Regret

- **No-swap regret:** for any swap function ϕ mapping original actions i to alternatives $i' = \phi(i)$, you do not regret making that swap

$$\frac{1}{T}\sum_{t} \ell_t^{i_t} \leq \frac{1}{T}\sum_{t} \ell_t^{\phi(i_t)} + \tilde{\epsilon}(T,\delta), \qquad \text{w.p. } 1-\delta$$

- **Equivalently:** for **subset of periods** when you played i you don't regret any other action i'

$$\frac{1}{T}\sum_{t:i_t=i} \ell_t^{i_t} \leq \max_{i'} \frac{1}{T}\sum_{t:i_t=i} \ell_t^{i'} + \tilde{\epsilon}(T,\delta), \qquad \text{w.p. } 1-\delta$$

You have an online learning problem, for simplicity, with 2 actions. Is any no-swap regret sequence a no-regret sequence?

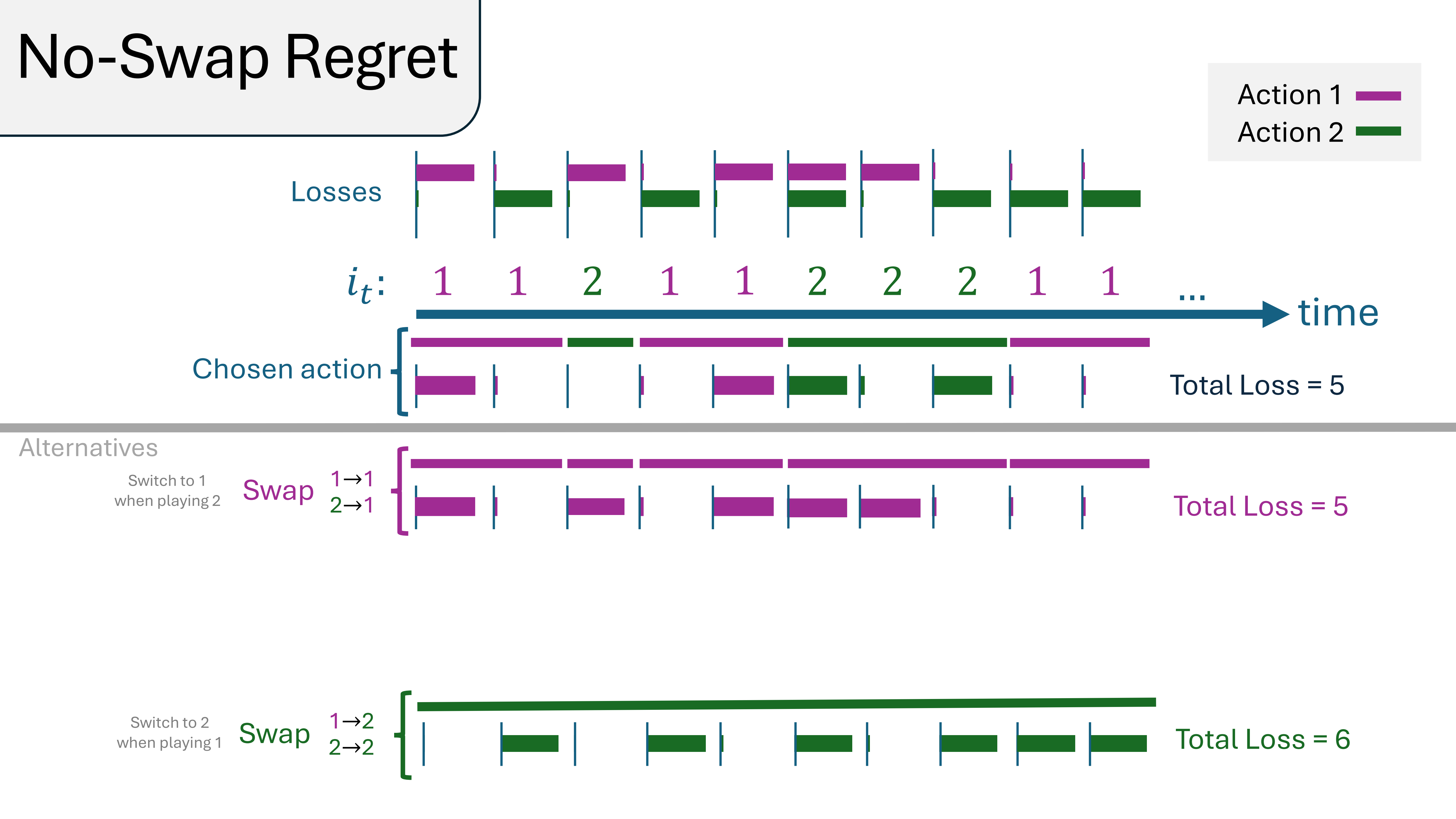
Yes		0%
No		0%

You have an online learning problem, for simplicity, with 2 actions. Is any no-regret sequence a no-swap regret sequence?

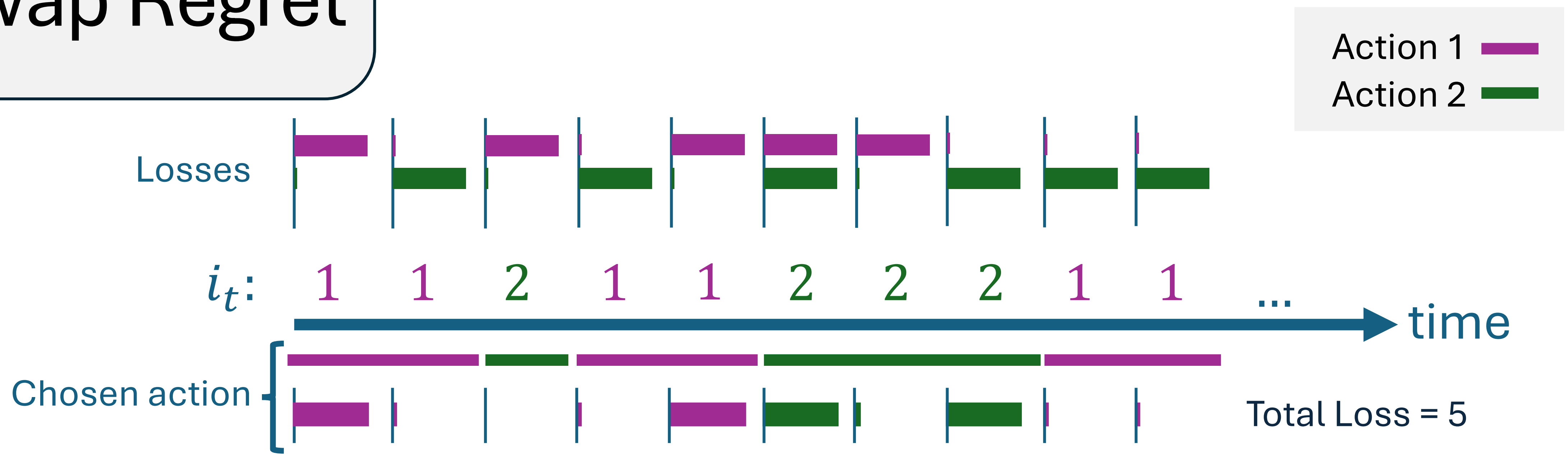
Yes		0%
No		0%

You have an online learning problem, for simplicity, with 3 actions. Is any no-regret sequence a no-swap regret sequence?

Yes		0%
No		0%

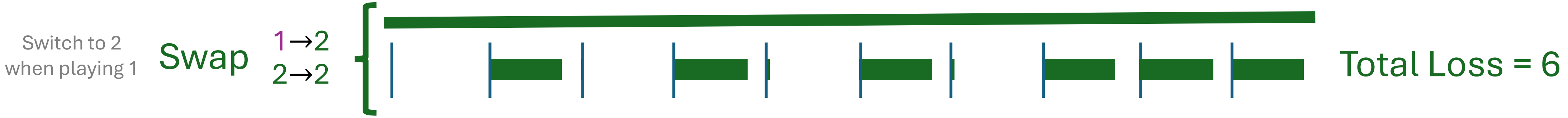

No-Swap Regret
Action 1
Action 2
Losses
i_t: 1 1 2 1 1 2 2 2 1 1 ...
time
Chosen action
Total Loss = 5
Alternatives
Switch to 1 when playing 2
Swap
1→1
2→1
Total Loss = 5
Switch to 2 when playing 1
Swap
1→2
2→2
Total Loss = 6

No-Swap Regret



Alternatives

No-swap regret is weirdly implied by no-regret when you only have two actions.
Intuition: no-regret towards action j is the same as no-regret on the subset of periods when you did not play j. With two actions, these are exactly the periods when you played j'



Can we reduce no-swap regret to no-regret?

No Swap Regret vs No Regret

- **For subset of periods** when played i don't regret any other i'

$$\frac{1}{T}\sum_{t:i_t=i}\ell_t^{i_t} \leq \max_{i'}\frac{1}{T}\sum_{t:i_t=i}\ell_t^{i'} + \tilde{\epsilon}(T,\delta), \qquad \text{w.p. } 1-\delta$$

- This looks like the no-regret property, but on a subset of periods
- If ahead of time we knew on which subset of periods we'd play i
- We could spawn a separate no-regret algorithm A_i
- When it was time to play i we would call A_i and report back loss

Swap to No-Regret Reduction

actions

1

⋮

j

⋮

n

Master Algorithm (M)

A_1
Responsible for controlling regret in periods when 1 was played

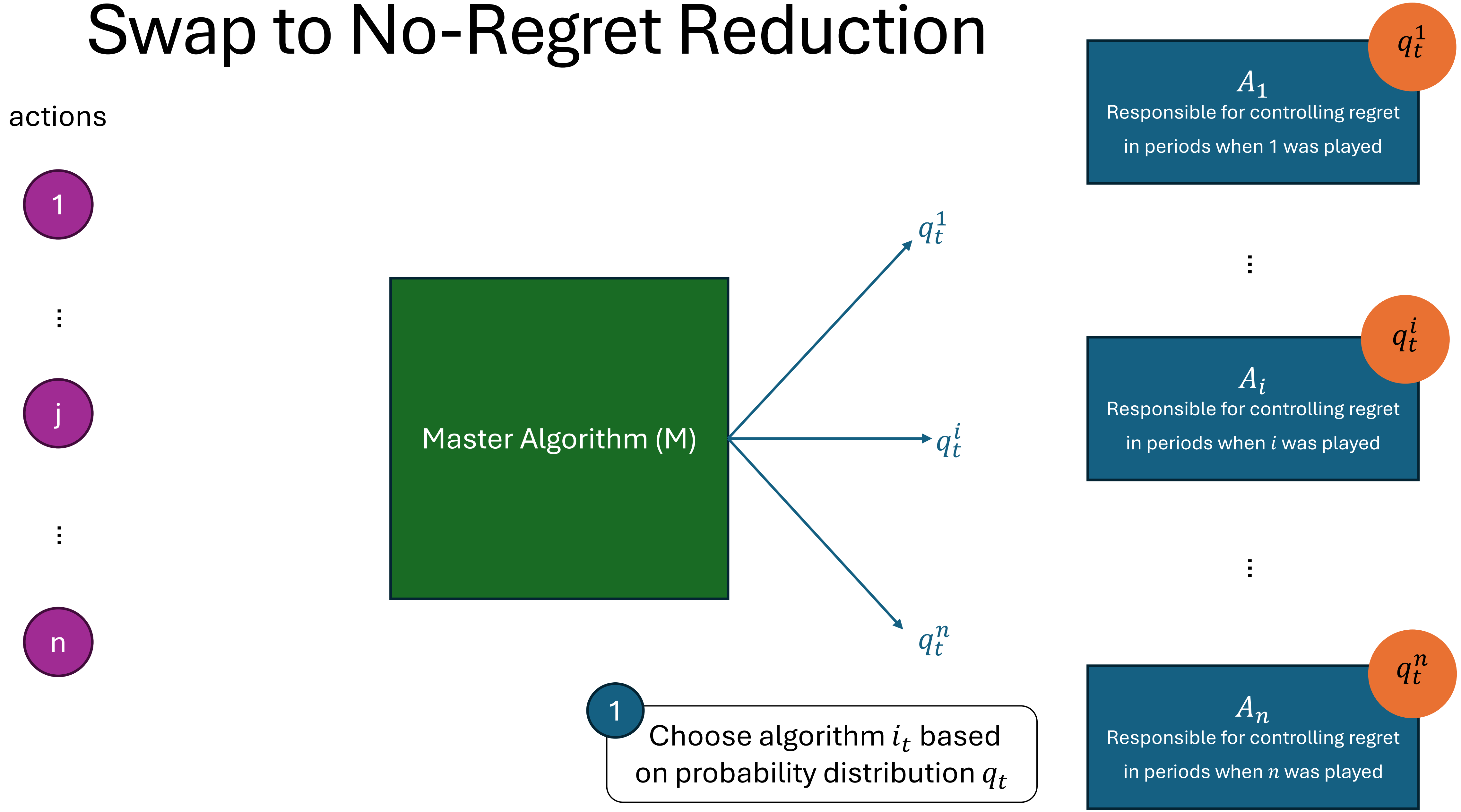
⋮

A_i
Responsible for controlling regret in periods when i was played

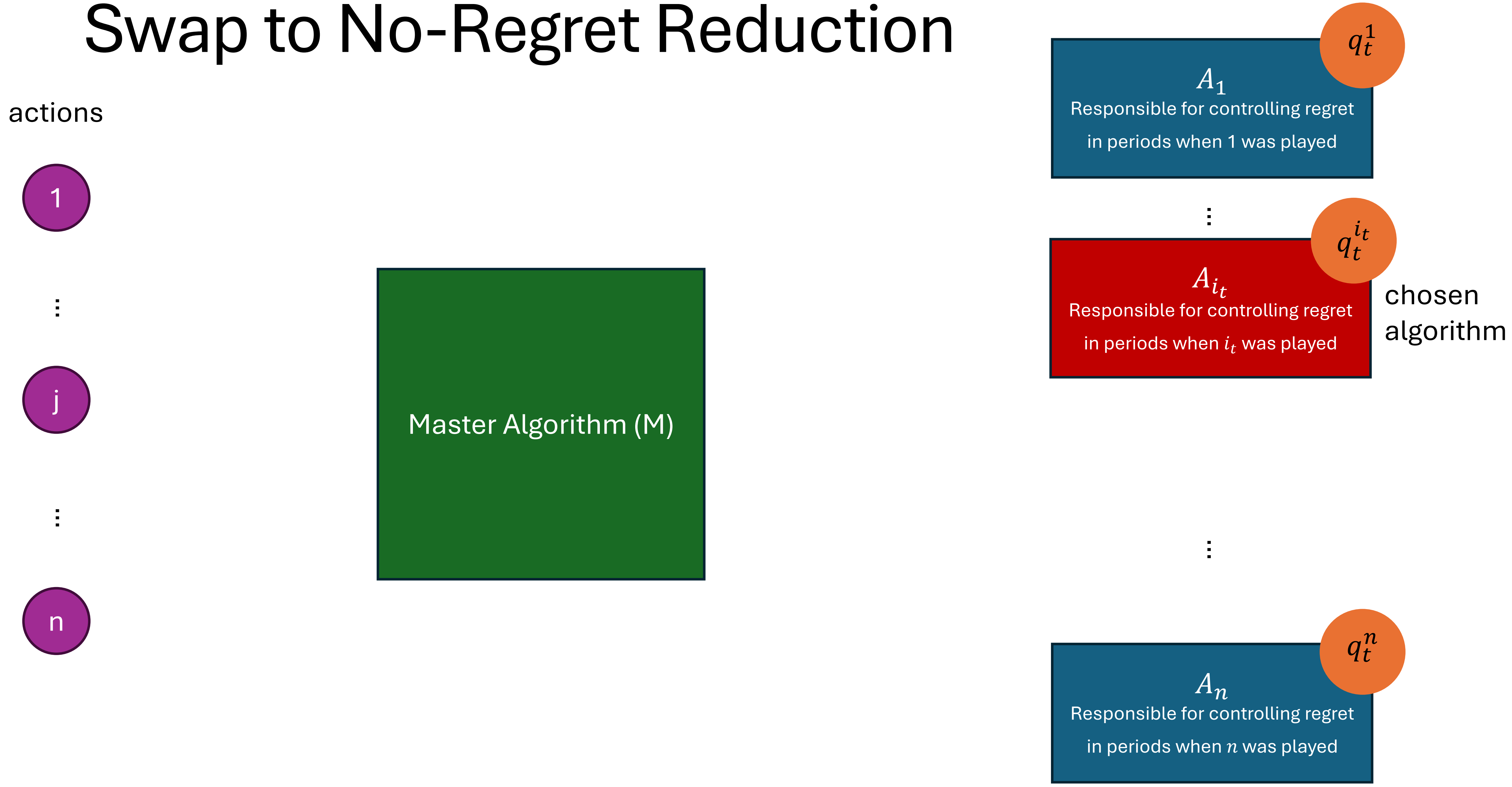
⋮

A_n
Responsible for controlling regret in periods when n was played

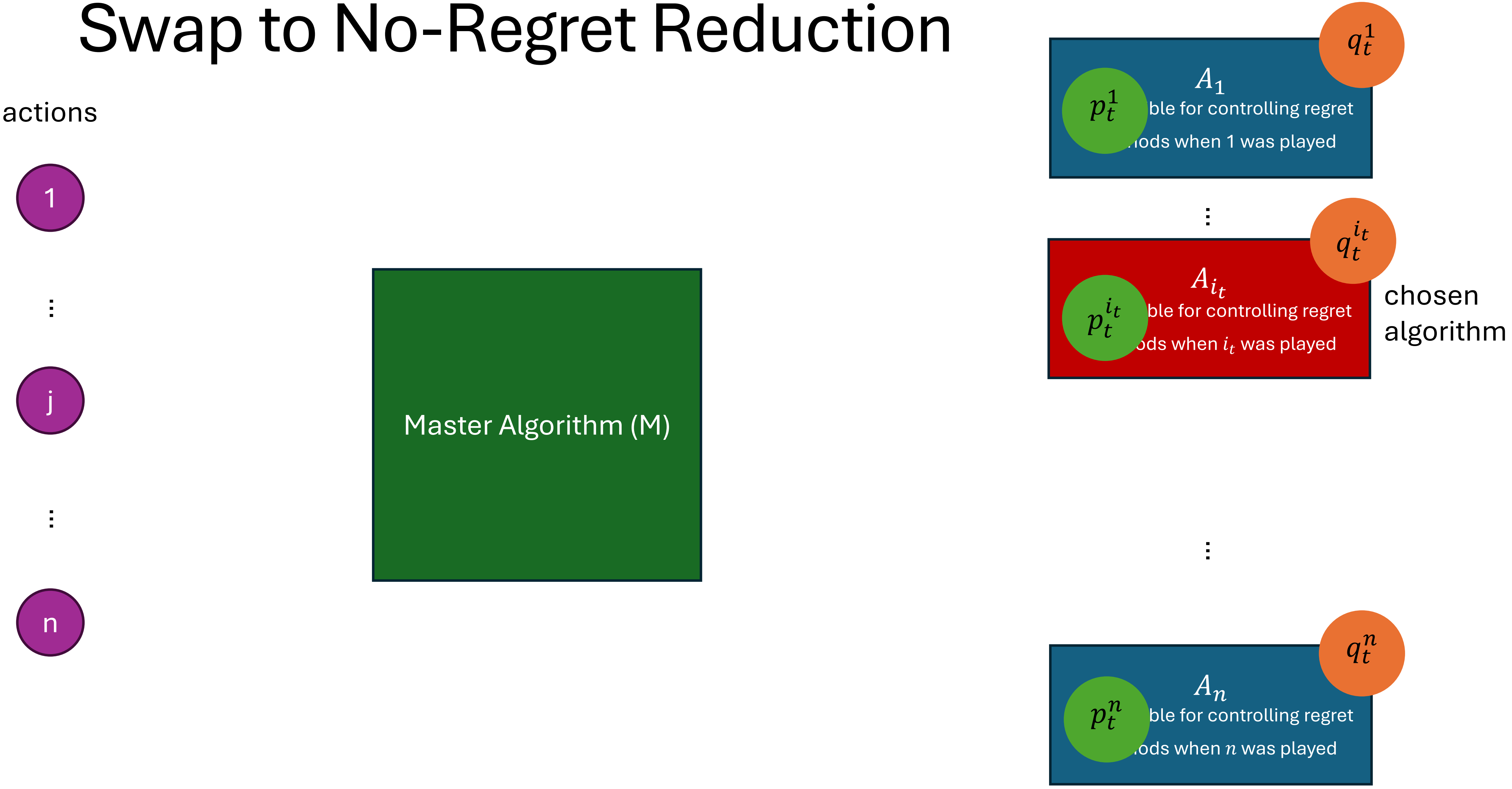
Swap to No-Regret Reduction



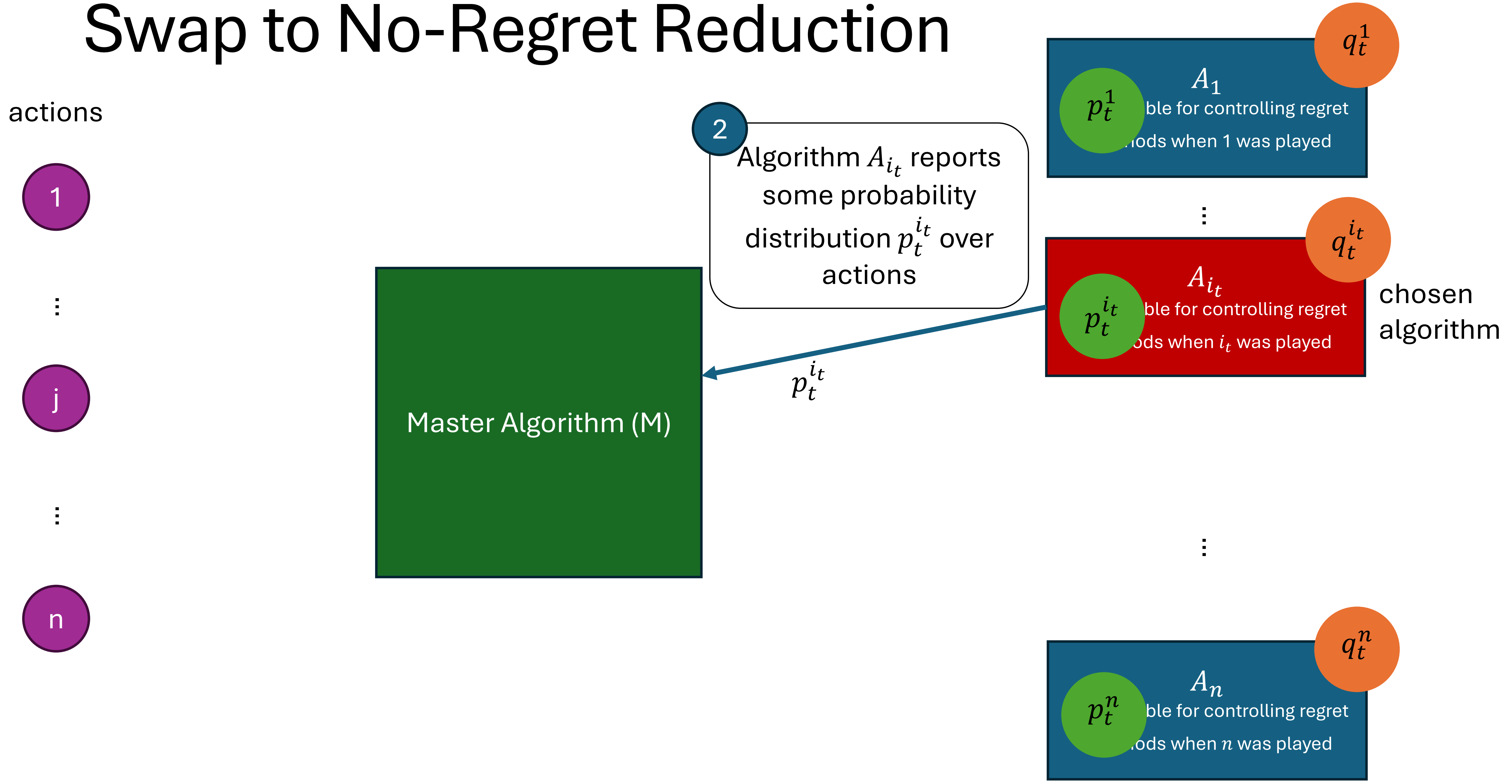
Swap to No-Regret Reduction



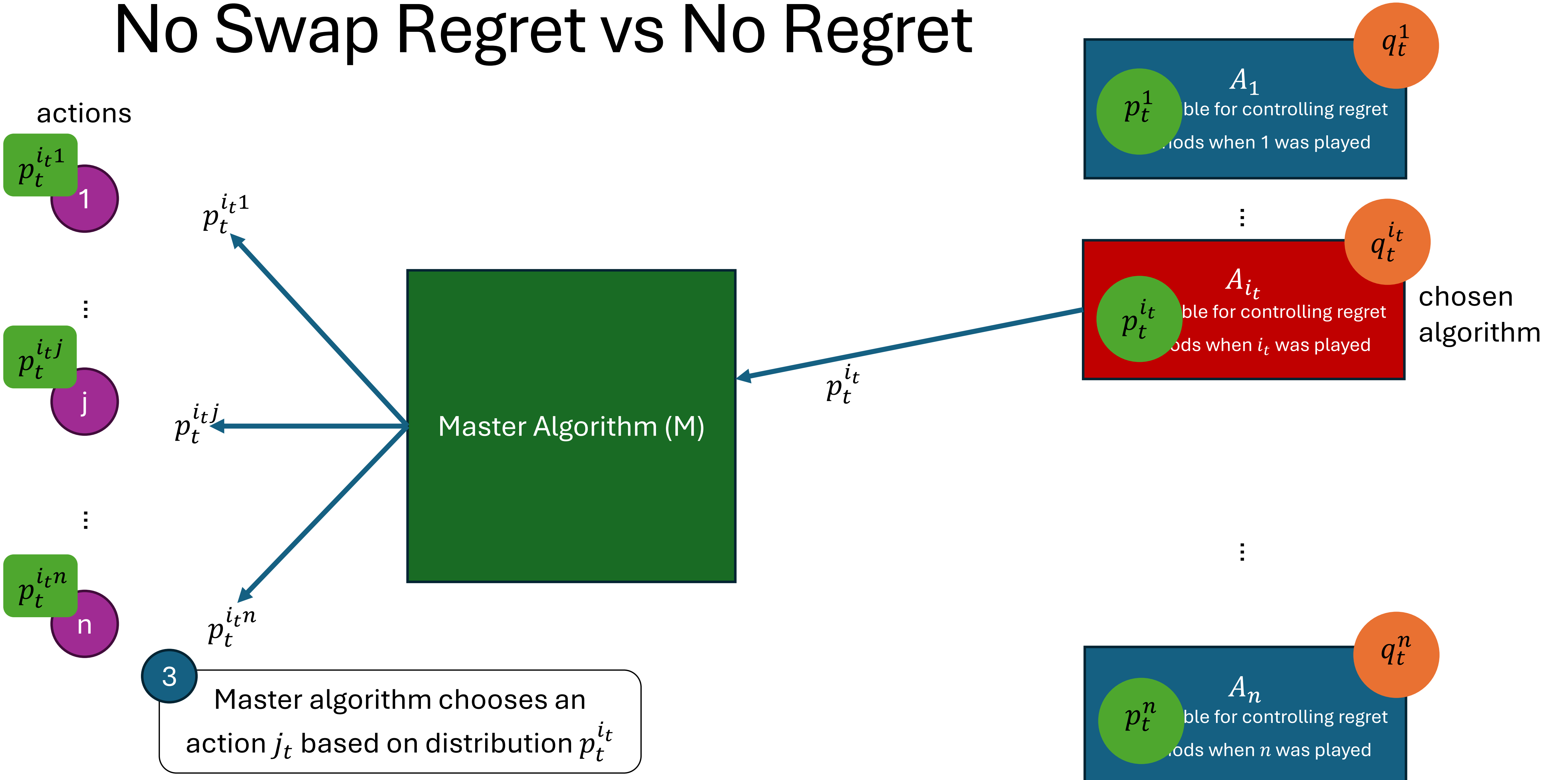
Swap to No-Regret Reduction



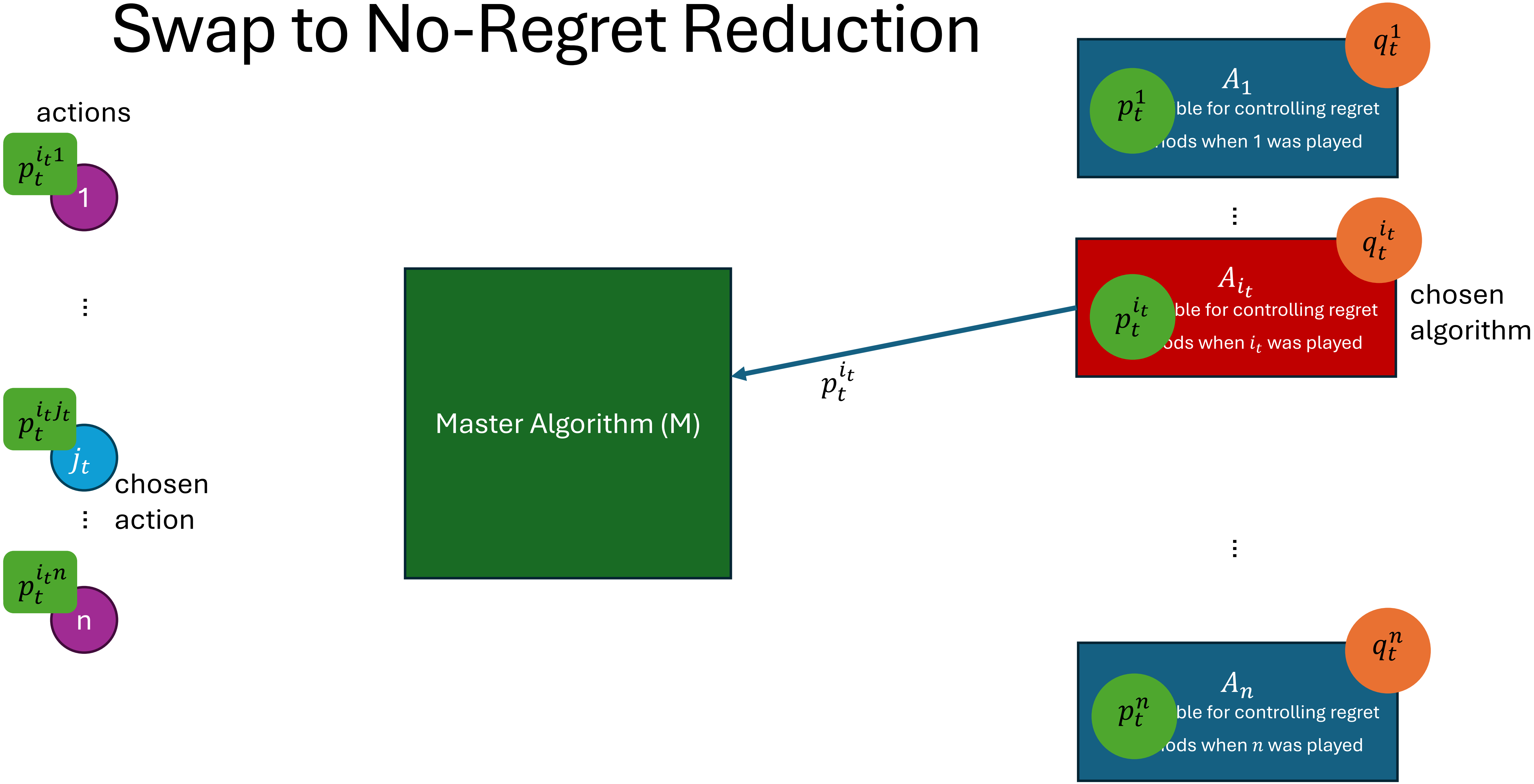
Swap to No-Regret Reduction



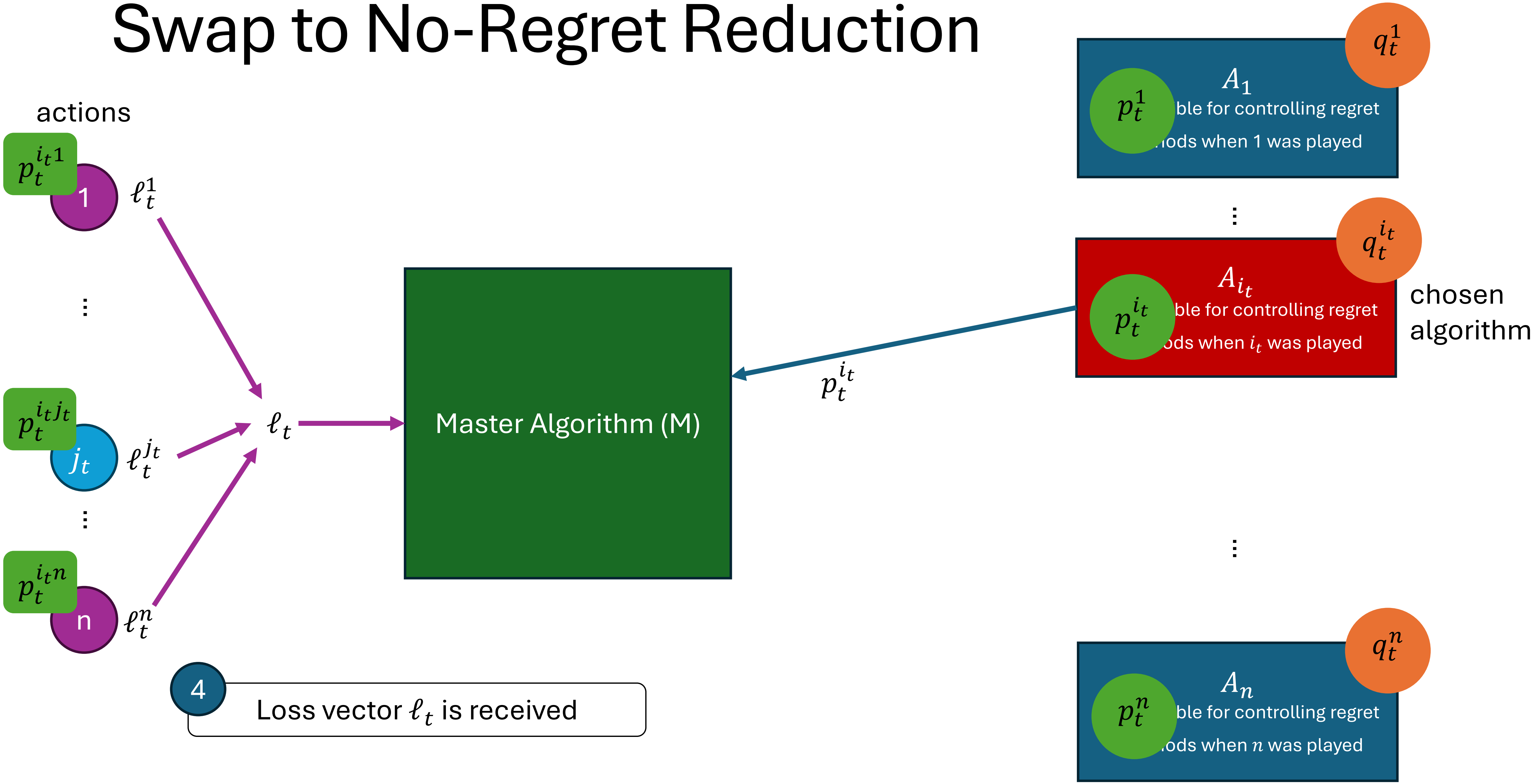
No Swap Regret vs No Regret



Swap to No-Regret Reduction

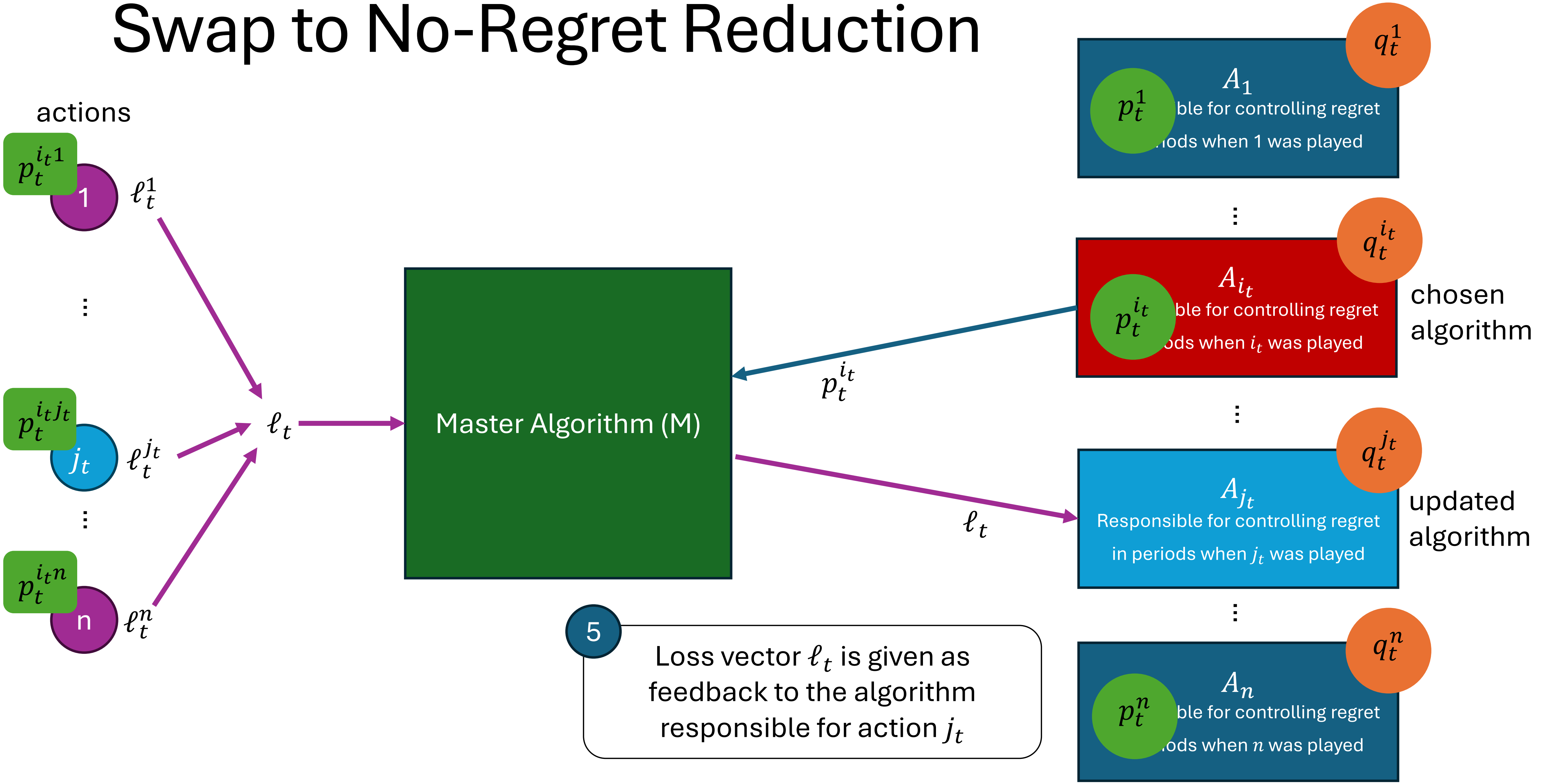


Swap to No-Regret Reduction



4 Loss vector ℓ_t is received

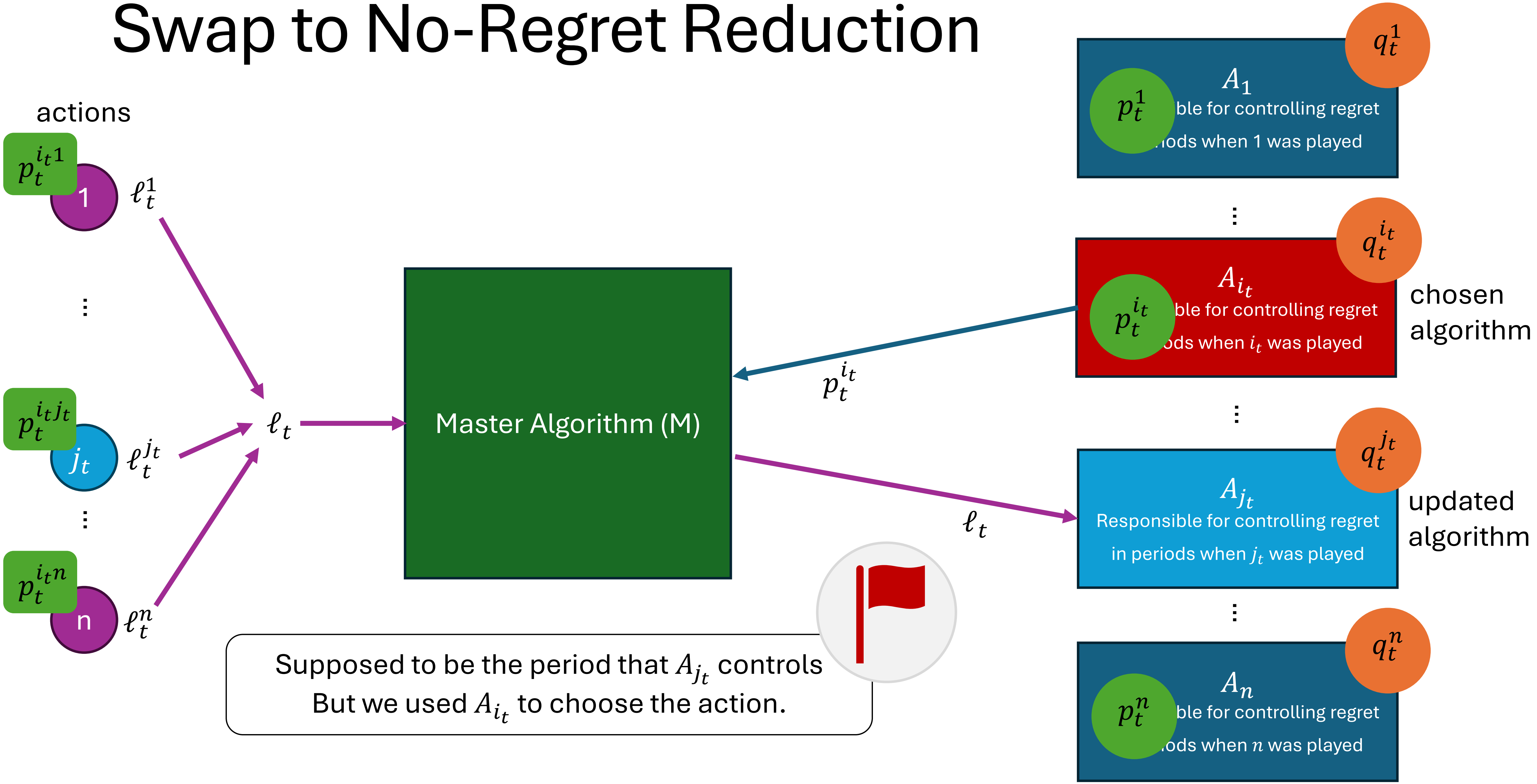
Swap to No-Regret Reduction



5

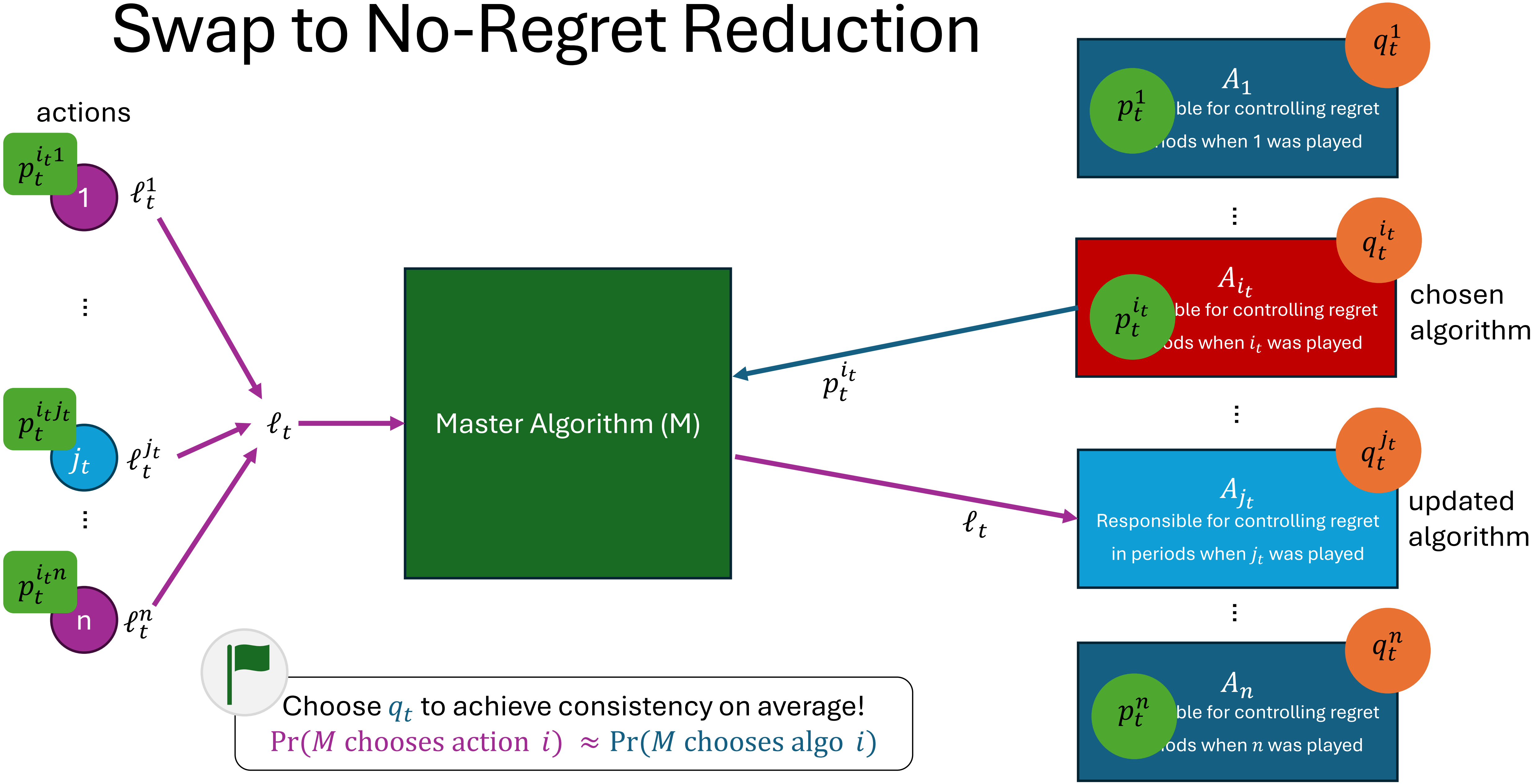
Loss vector ℓ_t is given as feedback to the algorithm responsible for action j_t

Swap to No-Regret Reduction



Supposed to be the period that A_{j_t} controls
But we used A_{i_t} to choose the action.

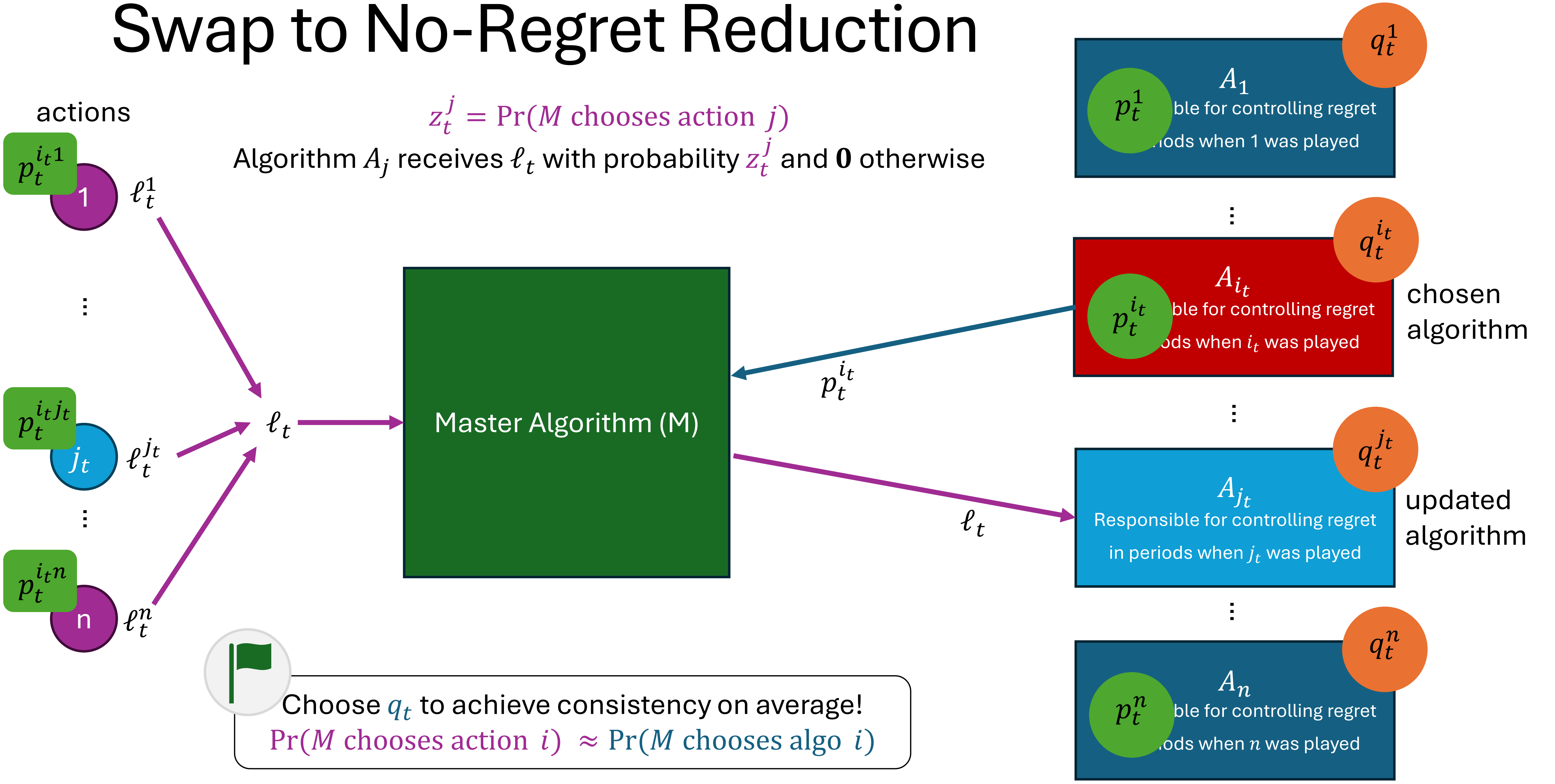
Swap to No-Regret Reduction



Choose q_t to achieve consistency on average!

$\Pr(M \text{ chooses action } i) \approx \Pr(M \text{ chooses algo } i)$

Swap to No-Regret Reduction



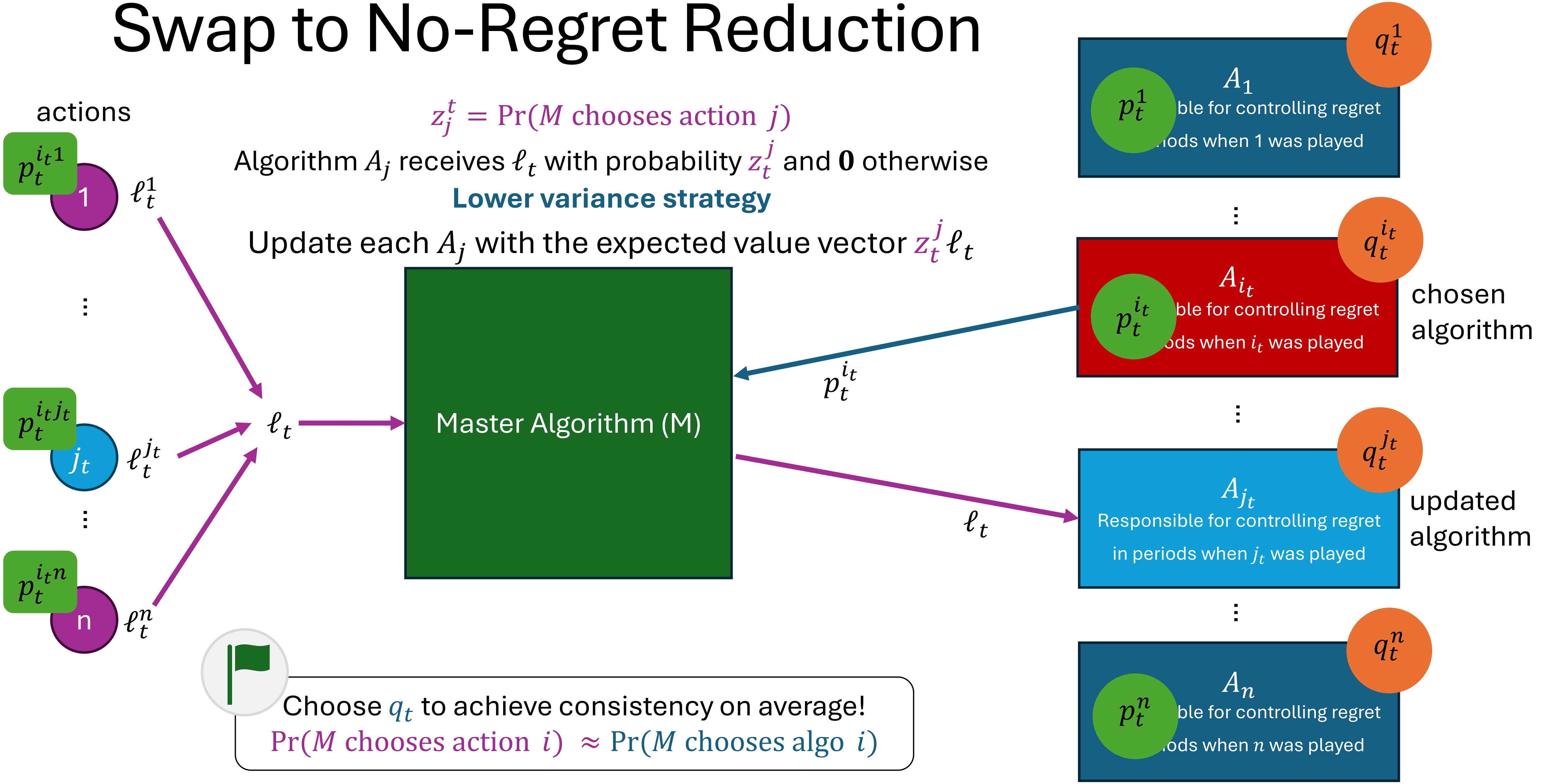
Swap to No-Regret Reduction

$z_j^t = \Pr(M \text{ chooses action } j)$

Algorithm A_j receives ℓ_t with probability z_t^j and $\mathbf{0}$ otherwise

Lower variance strategy

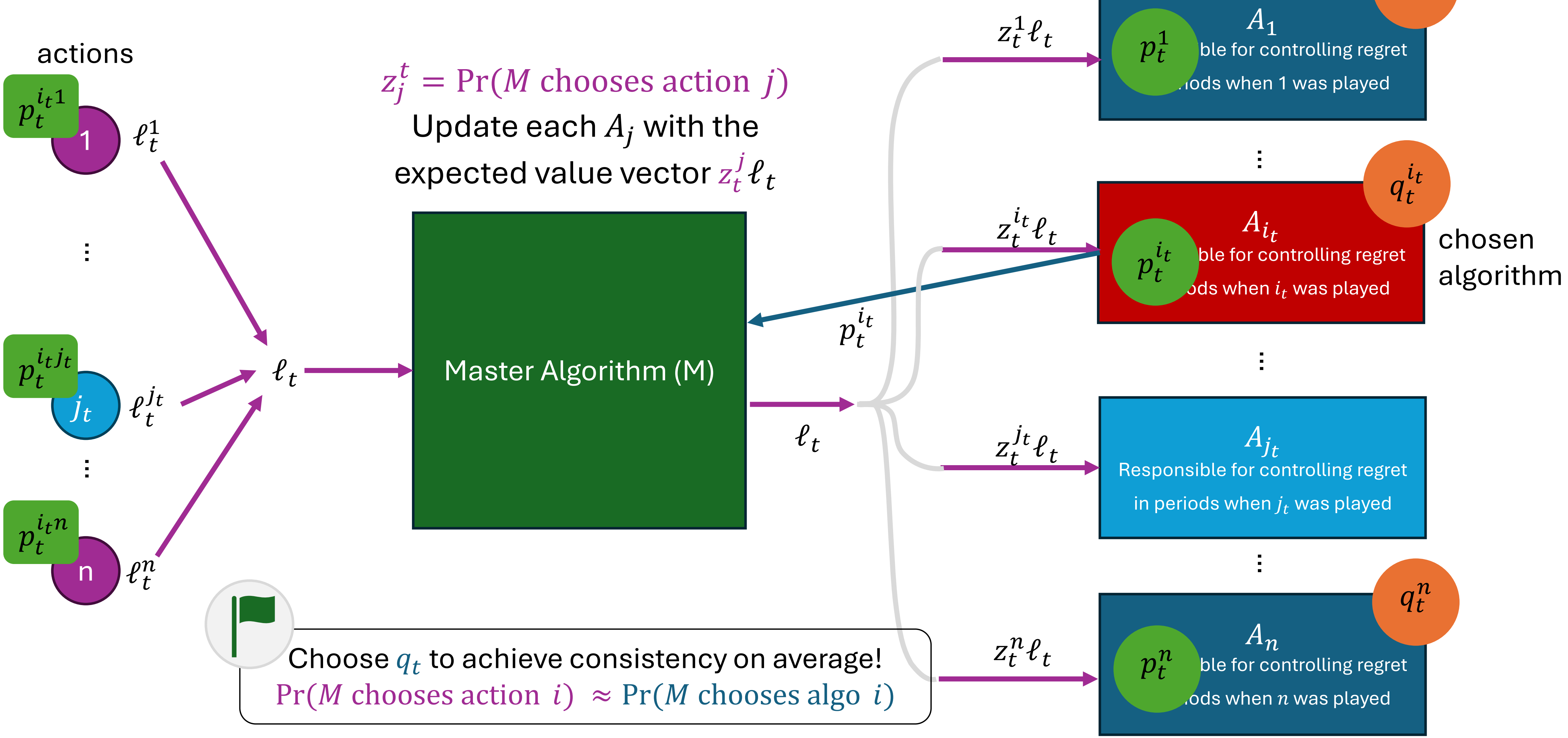
Update each A_j with the expected value vector $z_t^j \ell_t$



Choose q_t to achieve consistency on average!

$\Pr(M \text{ chooses action } i) \approx \Pr(M \text{ chooses algo } i)$

Sum: The Reduction Protocol



Sum: The reduction protocol

- At each period we choose each action with probability

$$z_t^j = \Pr(M \text{ choose action } j)$$
$$= \sum_i \underbrace{\Pr(M \text{ choose algo } A_i)}_{q_t^i} \cdot \underbrace{\Pr(A_i \text{ choose action } j)}_{p_t^{ij}}$$

- We update each algorithm A_j with loss vector

$$z_t^j \ell_t = \Pr(M \text{ choose action } j) \cdot (\text{loss vector})$$

- The distribution over algorithms q_t is chosen such that

$$\Pr(M \text{ choose action } j) \approx \Pr(M \text{ choose algo } A_j)$$

From No-Regret of Algos
to No-Swap Regret of Master

Regret = Loss – Benchmark Loss

Loss Analysis at Each Step

- How much loss does algorithm A_i perceive?

$$\underbrace{\Pr(M \text{ choose action } i)}_{\text{The fraction of the loss vector that M attributed and reported back to } A_i} \sum_j \Pr(A_i \text{ choose action } j) \cdot \text{loss}(j)$$

- How much total loss do all the algorithms perceive?

$$\sum_i \Pr(M \text{ choose action } i) \sum_j \Pr(A_i \text{ choose action } j) \cdot \text{loss}(j)$$

- How much loss does the master algorithm incur?

$$\sum_i \Pr(M \text{ choose algo } A_i) \sum_j \Pr(A_i \text{ choose action } j) \cdot \text{loss}(j)$$

Loss Analysis at Each Step

- How much loss does algorithm A_i perceive?

$$\underbrace{\Pr(M \text{ choose action } i)}_{\text{The fraction of the loss vector that M attributed and reported back to } A_i} \sum_j \Pr(A_i \text{ choose action } j) \cdot \text{loss}(j)$$

- How much total loss do all the algorithms perceive?

$$\sum_i \Pr(M \text{ choose action } i) \sum_j \Pr(A_i \text{ choose action } j) \cdot \text{loss}(j)$$

- How much loss does the master algorithm incur?

$$\sum_i \Pr(M \text{ choose algo } A_i) \sum_j \Pr(A_i \text{ choose action } j) \cdot \text{loss}(j)$$

Recap: Loss Analysis at Each Step

Corollary. If we can guarantee that

$$\underbrace{\Pr(M \text{ choose action } i)}_{z_t^i} \approx \underbrace{\Pr(M \text{ choose algo } A_i)}_{q_t^i}$$

Then the total loss perceived by the separate algorithms is approximately the same as the total loss experienced by the master

total loss perceived by algos $\approx$ total loss of master

Competing Benchmark Analysis at Each Step

- What can each algorithm A_i compete with based on **no-regret**?

$$\underbrace{\Pr(M \text{ choose action } i)}_{\text{The fraction of the loss vector that M attributed and reported back to } A_i} \cdot \text{loss}\big(\underbrace{\phi(i)}_{\text{For each algo } A_i \text{ this is a constant action comparison with } i' = \phi(i)}\big)$$

- What can in total all algorithms compete with based on **no-regret**?

$$\sum_i \Pr(M \text{ choose action } i) \cdot \text{loss}\big(\phi(i)\big)$$

- What does the master want to compete with for **no-swap regret**?

$$\sum_j \Pr(M \text{ choose action } j) \cdot \text{loss}\big(\phi(j)\big)$$

Competing Benchmark Analysis at Each Step

- What can each algorithm A_i compete with based on **no-regret**?

$$\text{Pr}(M \text{ choose action } i) \cdot \text{loss}\big(\phi(i)\big)$$

The fraction of the loss vector that M attributed and reported back to A_i

For each algo A_i this is a constant action comparison with $i' = \phi(i)$

- What can in total all algorithms compete with based on **no-regret**?

$$\sum_i \text{Pr}(M \text{ choose action } i) \cdot \text{loss}\big(\phi(i)\big)$$

$$=$$

$$\sum_j \text{Pr}(M \text{ choose action } j) \cdot \text{loss}\big(\phi(j)\big)$$

- What does the master want to compete with for **no-swap regret**?

Recap: Benchmark Analysis at Each Step

Corollary. The total *perceived* benchmark loss that algorithms compete with, *where each algorithm* i *considers the no-regret benchmark of always playing action* $i' = \phi(i)$, is equal to the *true* swap benchmark loss that the master wants to compete with, *associated with the swap function* ϕ.

Regret = Loss – Benchmark Loss

Regret Analysis at Each Step

Corollary. If we can guarantee that

$$\Pr(M \text{ choose action } i) \approx \Pr(M \text{ choose algo } A_i)$$

then swap regret of master is upper bounded by sum of plain regrets of algos

$$\begin{aligned}
\text{Swap Regret of Master } &= \text{Total Loss of Master} - \text{Swap Benchmark} \\
&\approx \text{Total Perceived Loss by Algos} - \text{Total Algo Fixed Action Benchmark} \\
&= \text{Total Perceived Regret of Algos}
\end{aligned}$$

Regret Analysis at Each Step

Corollary. If we can guarantee that

$$\Pr(M \text{ choose action } i) \approx \Pr(M \text{ choose algo } A_i)$$

then swap regret of master is upper bounded by sum of plain regrets of algos

$$\sum_t \sum_j z_t^j \ell_t^j - z_t^j \ell_t^{\phi(j)} = \sum_t \sum_i \boldsymbol{q_t^i} \sum_j p_t^{ij} \ell_t^j - \sum_t \sum_j z_t^j \ell_t^{\phi(j)}$$

Swap Regret of Master — Total Loss of Master — Swap Benchmark

$$\approx \sum_t \sum_i \boldsymbol{z_t^i} \sum_j p_t^{ij} \ell_t^j - \sum_t \sum_i z_t^i \ell_t^{\phi(i)}$$

Total Loss Perceived by Algos — Total Algo Fixed Action Benchmark

$$= \sum_i \sum_t \langle p_t^i, z_t^i \ell_t \rangle - z_t^j \ell_t^{\phi(i)}$$

Total Perceived Regret of Algos

Can we pick q_t such that:

$$\Pr(M \text{ choose action } j) \approx \Pr\left(M \text{ choose algo } A_j\right)$$

Choosing distribution over algos

- Choose q_t such that

$$\Pr(M \text{ choose action } j) \approx \Pr\big(M \text{ choose algo } A_j\big)$$

- Remember that

$$\Pr(M \text{ choose action } j) = \sum_i \Pr(M \text{ choose algo } A_i) \cdot \Pr(A_i \text{ choose action } j)$$

- We need the distribution over algos q_t to satisfy the self-consistency property

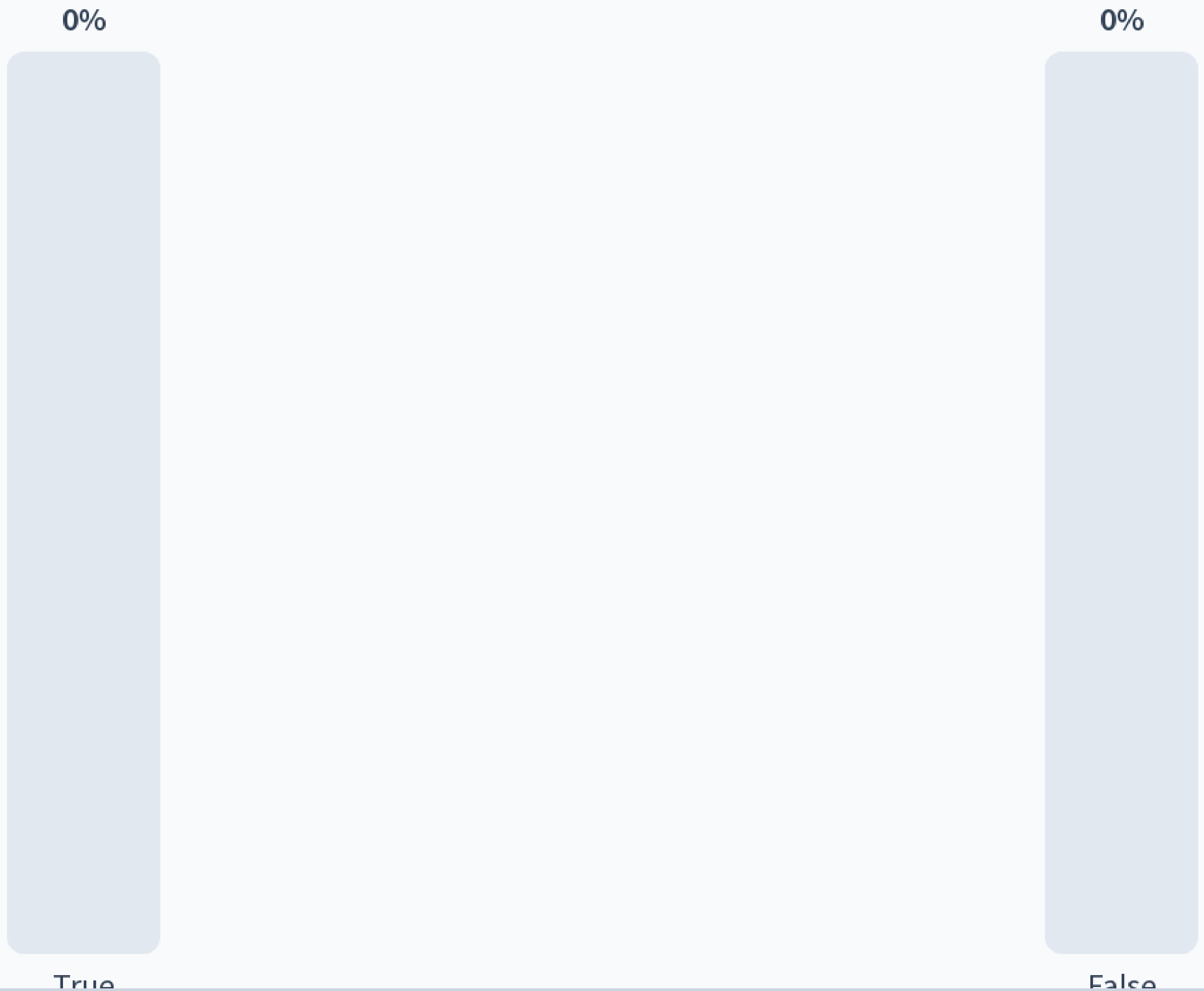
$$\sum_i \underbrace{\Pr(M \text{ choose algo } A_i)}_{q_t^i} \cdot \underbrace{\Pr(A_i \text{ choose action } j)}_{p_t^{ij}} = \underbrace{\Pr\big(M \text{ choose algo } A_j\big)}_{q_t^j}$$

Does there exist a distribution q_t such that:

$$\sum_i \Pr(M \text{ choose algo } A_i) \cdot \Pr(A_i \text{ choose action } j) = \Pr\left(M \text{ choose algo } A_j\right)$$

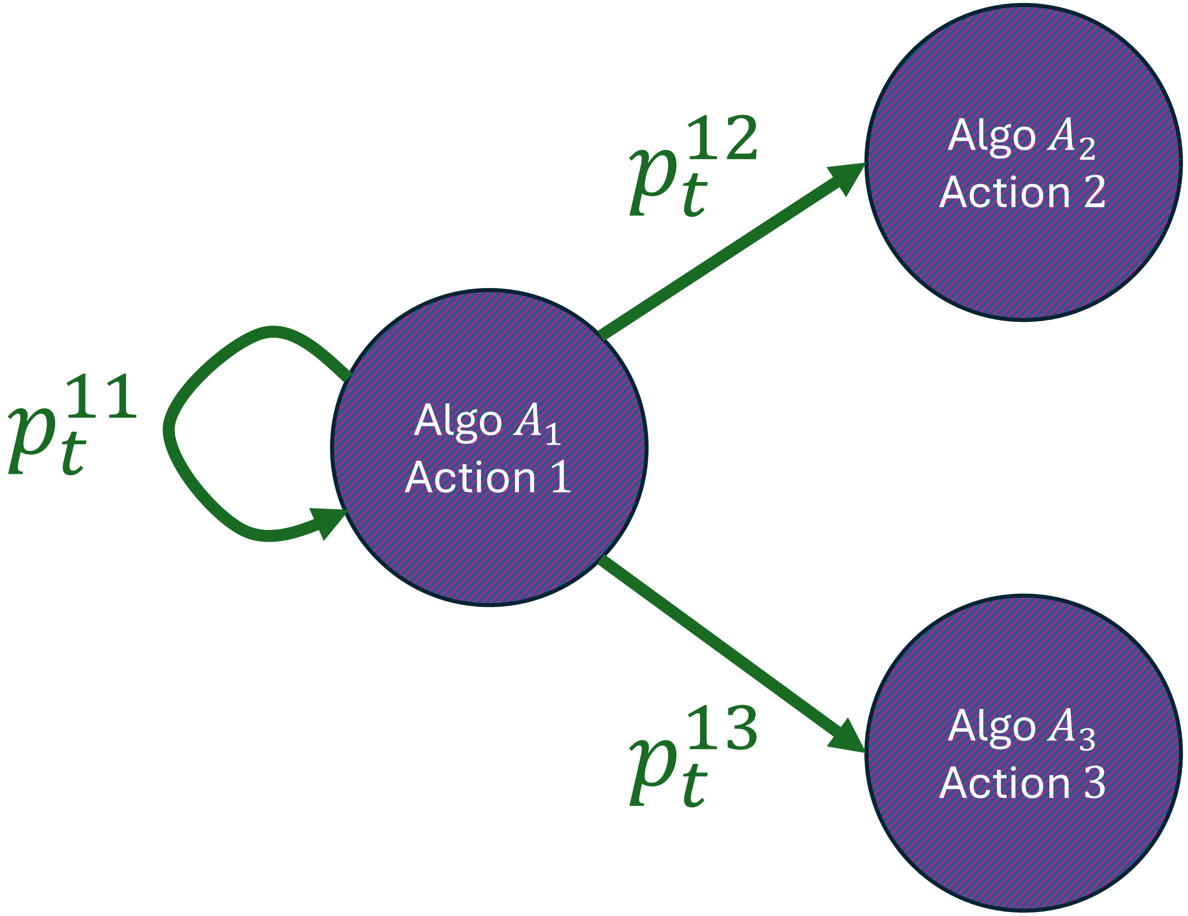
$$\sum_{i=1}^{n} q_t^i \cdot p_t^{ij} = q_t^j$$

There always exists a distribution q that satisfies this property



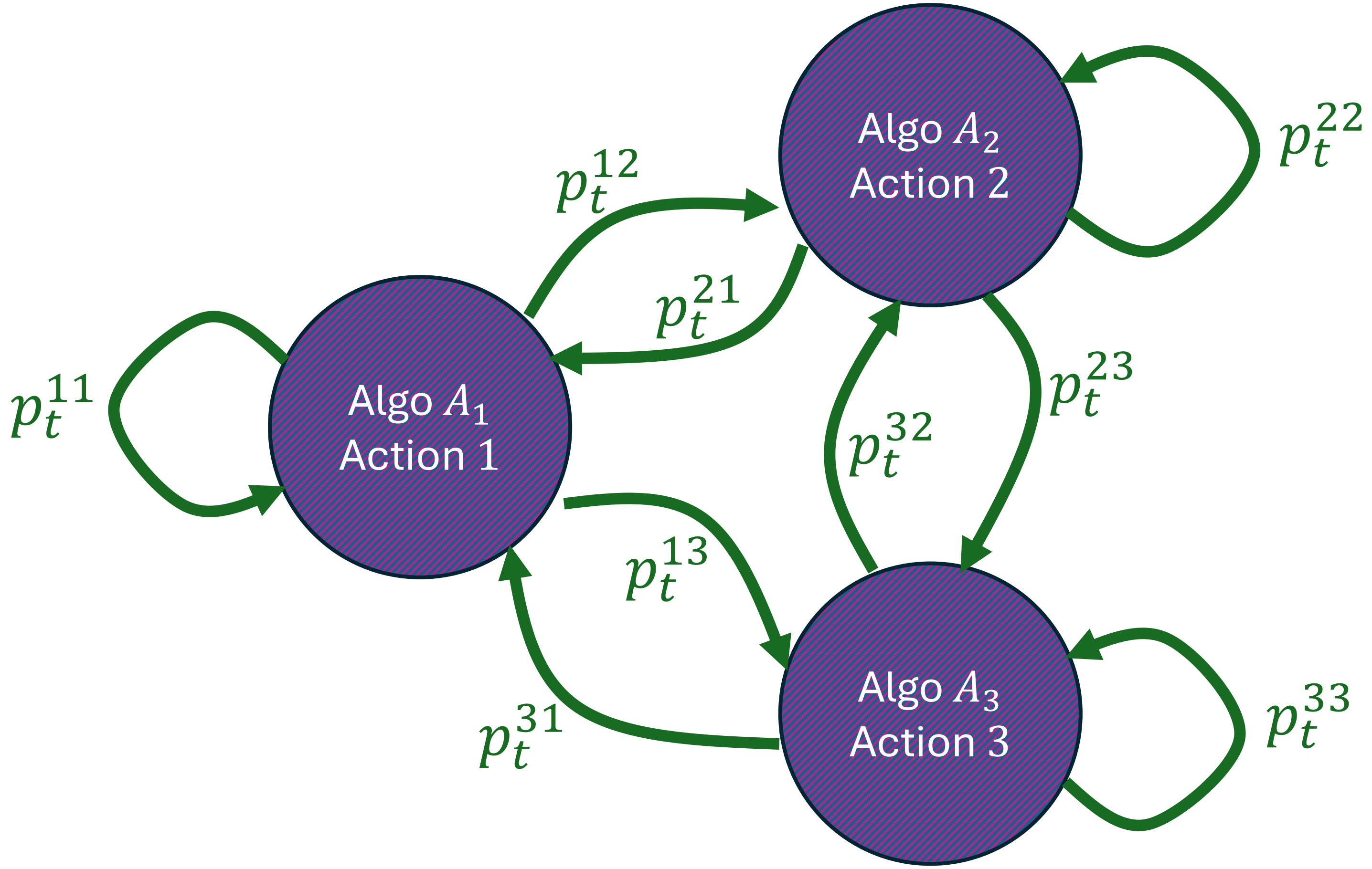
Choosing distribution over algos

$$\sum_i \Pr(M \text{ choose algo } A_i) \cdot \Pr(A_i \text{ choose action } j) = \Pr\big(M \text{ choose algo } A_j\big)$$



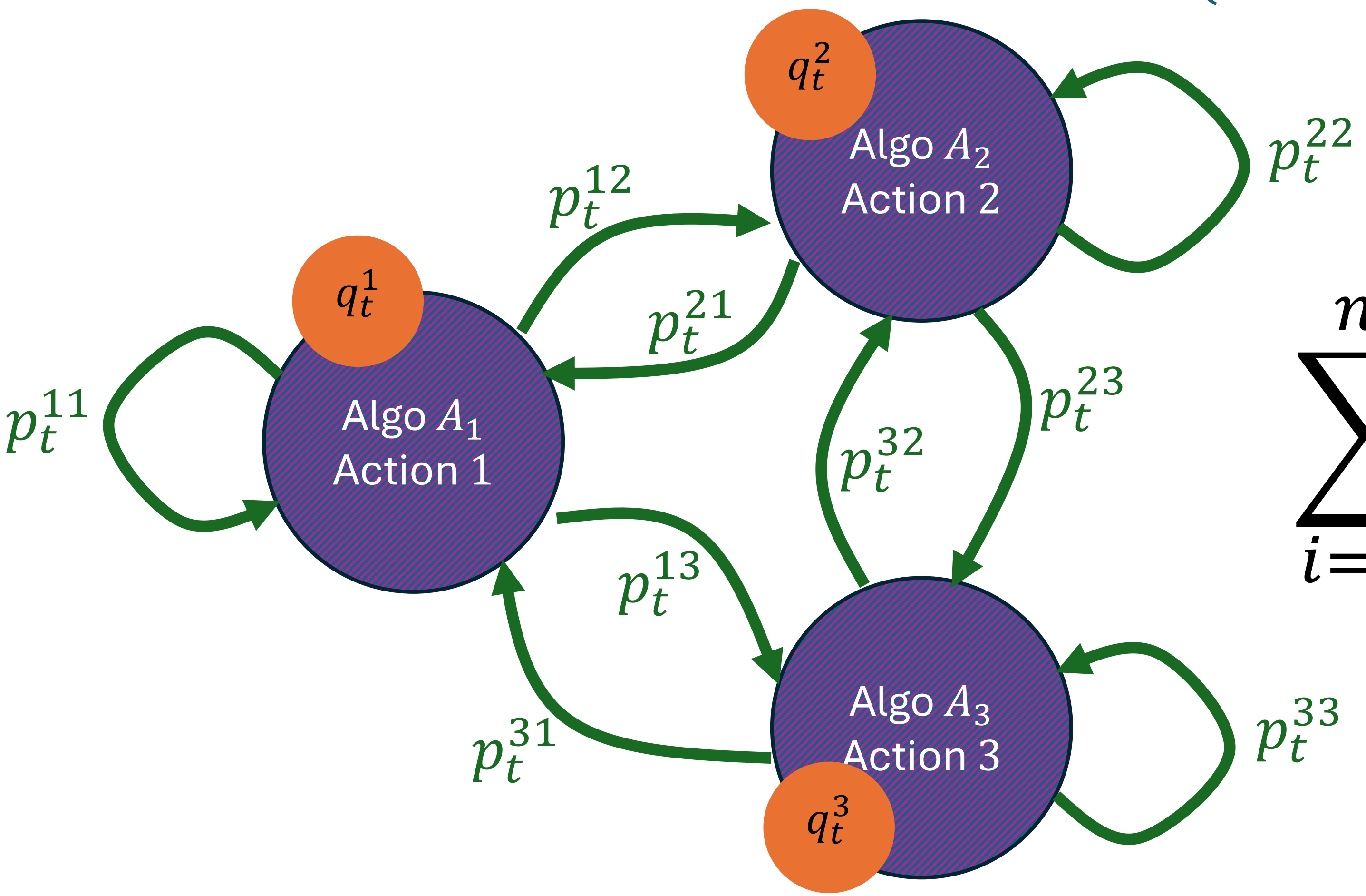
Choosing distribution over algos

$$\sum_i \Pr(M \text{ choose algo } A_i) \cdot \Pr(A_i \text{ choose action } j) = \Pr\big(M \text{ choose algo } A_j\big)$$



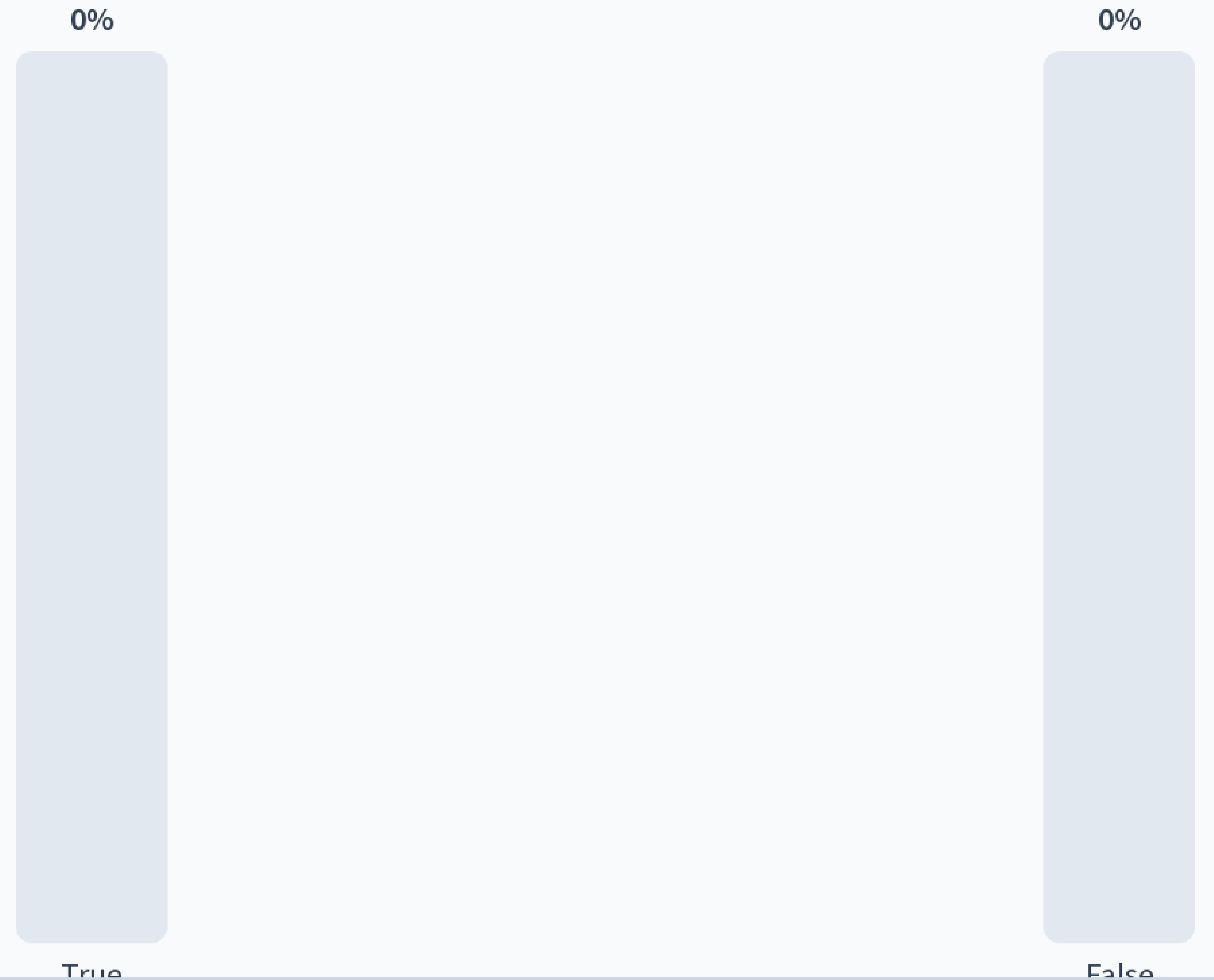
Choosing distribution over algos

$$\sum_i \Pr(M \text{ choose algo } A_i) \cdot \Pr(A_i \text{ choose action } j) = \Pr\big(M \text{ choose algo } A_j\big)$$



$$\sum_{i=1}^{n} q_t^i \cdot p_t^{ij} = q_t^j$$

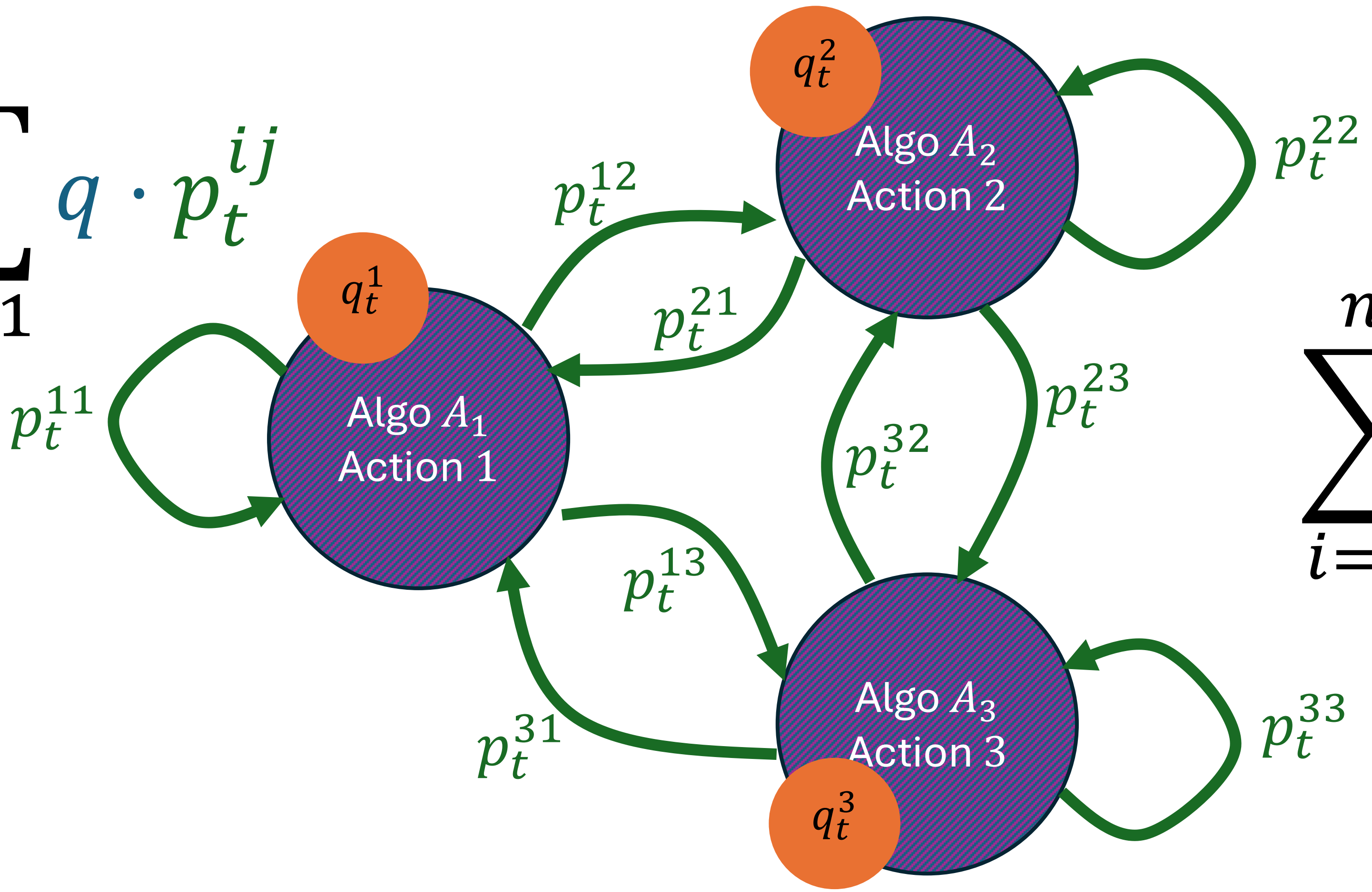
There always exists a distribution q that satisfies this property



A Markov Chain over the Algos/Actions

Starting from a distribution q over nodes and applying one step of the random transitions, brings us to a new distribution over states

$$q' = \sum_{i=1}^{n} q \cdot p_t^{ij}$$

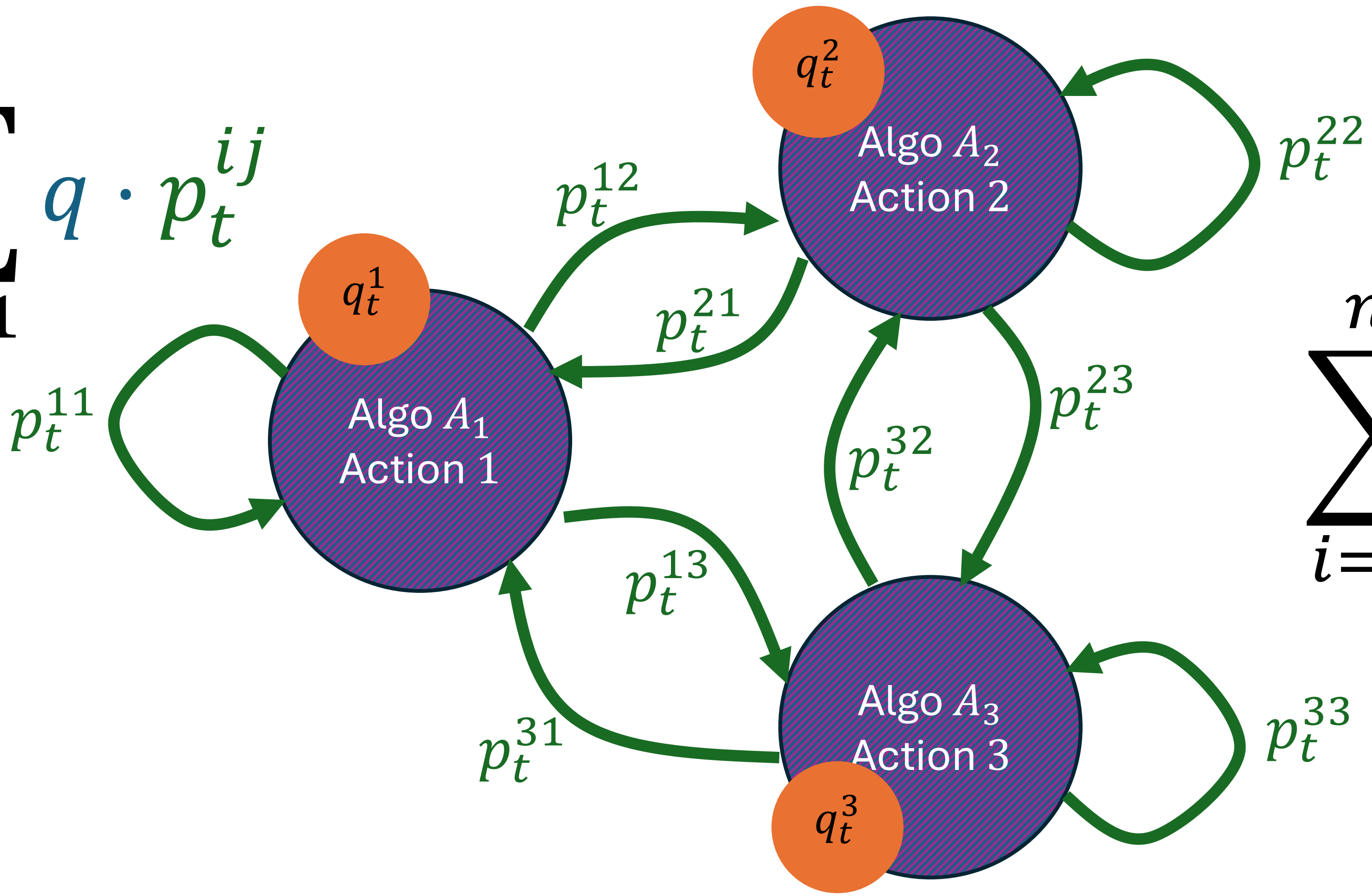


$$\sum_{i=1}^{n} q_t^i \cdot p_t^{ij} = q_t^j$$

Stationary Distributions of Markov Chains

If new distribution is the same as the original distribution, then this distribution is called a Stationary Distribution of the Markov Chain

$$q = \sum_{i=1}^{n} q \cdot p_t^{ij}$$

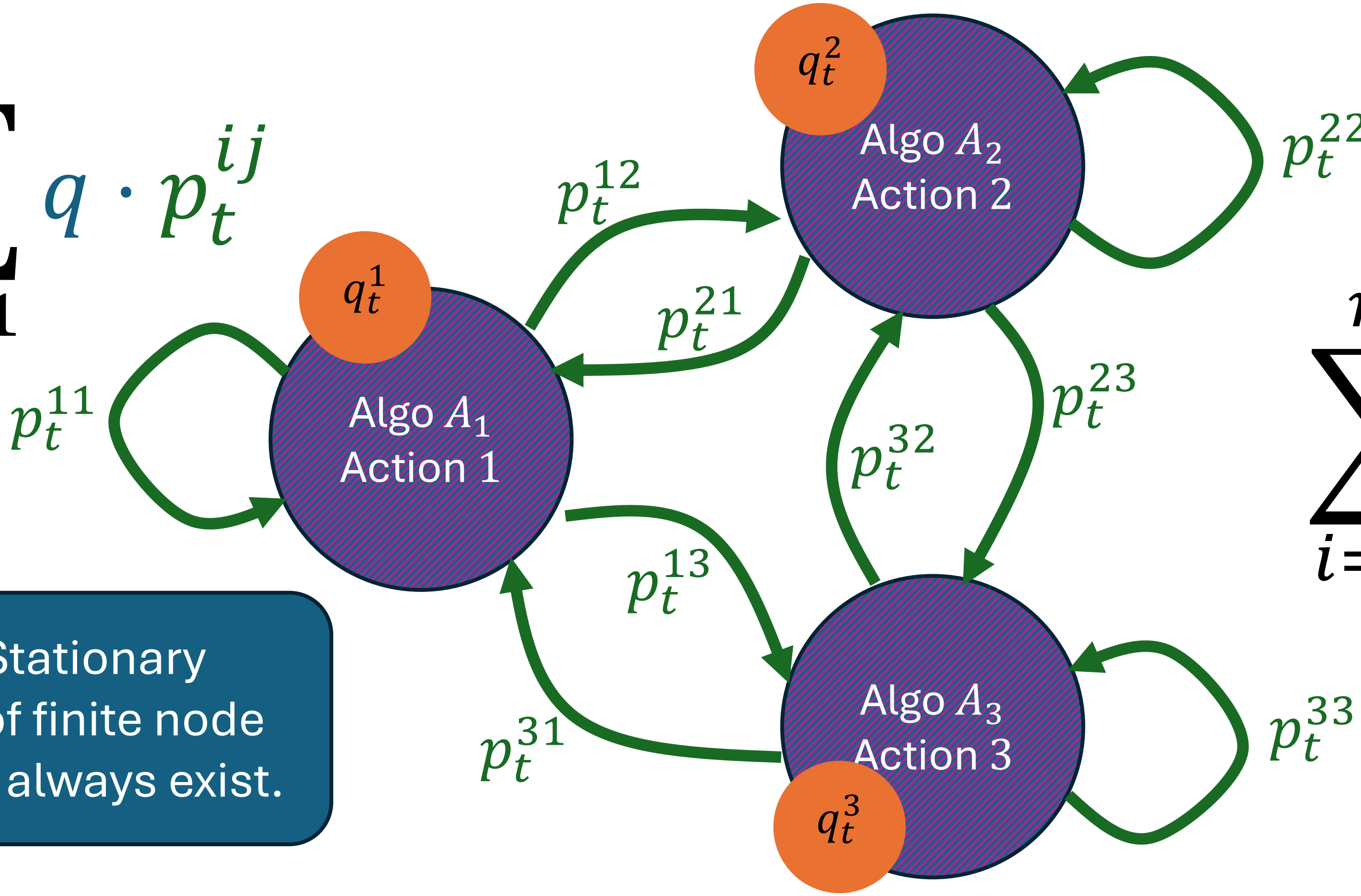


$$\sum_{i=1}^{n} q_t^i \cdot p_t^{ij} = q_t^j$$

Stationary Distributions of Markov Chains

If new distribution is the same as the original distribution, then this distribution is called a Stationary Distribution of the Markov Chain

$$q = \sum_{i=1}^{n} q \cdot p_t^{ij}$$



$$\sum_{i=1}^{n} q_t^i \cdot p_t^{ij} = q_t^j$$

Theorem. Stationary distributions of finite node Markov Chains always exist.

Recap: Choosing Distribution over Algos

Corollary. If we choose q_t as stationary distribution of the Markov Chain defined by transition probabilities $\Pr(\mathrm{i} \rightarrow \mathrm{j}) = p_t^{ij}$ then

$$\Pr(M \text{ choose action } j) = \Pr(M \text{ choose algo } A_j)$$

Therefore

$$\text{Swap Regret of Master} = \text{Total Fixed Action Regret of Algos} \rightarrow 0$$

Sum: The reduction protocol

- At each period calculate stationary distribution q_t of the Markov Chain defined by the transition probabilities $\Pr(i \to j) = p_t^{ij}$

- Choose each action with probability

$$z_t^j = \Pr(M \text{ choose action } j) = \Pr(M \text{ choose algo } j) = q_t^j$$

- Update each algorithm A_j with loss vector

$$z_t^j \ell_t = \Pr(M \text{ choose action } j) \cdot (\text{loss vector})$$

Finding Stationary Distributions

- Define the matrix P_t, whose (i, j) entry is p_t^{ij}
- Then the stationary distribution satisfies
$$q^\top = q^\top P_t$$
- q is a left eigenvector of P_t associated with eigenvalue 1
- We can calculate q via eigen-decomposition of P_t and identifying the eigenvector associated with eigenvalue 1

Overall Algorithm using EXP for each Algo

```
Initialize Pt with each row being the uniform distribution
For t in 1..T
    # Calculate choice probability q of master based on
    # choice probabilities Pt of algos
    Calculate stationary distribution q of matrix Pt
    Draw action jt based on distribution q
    Observe loss vector lt

    # update each algorithms choice probabilities
    For i in 1..n
        Calculate perceived loss plt[i] = q[i] * lt
        Pt[i] = EXP-Update(Pt[i], plt[i])
```

Recap: Final Theorem

Theorem. If we choose q_t as stationary distribution of the Markov Chain defined by transition probabilities $\Pr(\mathrm{i} \to \mathrm{j}) = p_t^{ij}$ and each algorithm updates their choice probabilities using the EXP rule then

$$\text{Average Swap Regret of Master} \leq 2n\sqrt{\frac{\log(n)}{T}} \to 0$$

Back to Games

Convergence to Correlated Equilibrium

Theorem. If all players use such an algorithm, then the empirical joint distribution of actions converges to the set of correlated equilibria.

At every T the empirical joint distribution of strategies π^T is an $\epsilon(T)$ approximate correlated equilibrium, in the sense that:

$$\text{SwapRegret}_i(s_i, s_i', T) = \boxed{\sum_{s_{-i}} \pi^T(s_i, s_{-i}) \cdot \left(u_i(s_i', s_{-i}) - u_i(s_i, s_{-i})\right)} \leq \epsilon(T)$$

with $\epsilon(T) = 2n\sqrt{\frac{\log(n)}{T}}$, where n is number of actions of player i

Expected gains from switching to s_i' whenever you played s_i

Note on Approximation Error

$$\sum_{s_{-i}} \pi^T(s_i, s_{-i}) \cdot \left(u_i(s_i', s_{-i}) - u_i(s_i, s_{-i})\right) \leq \epsilon$$

- If we wanted to analyze the conditional expectation of gains:

$$E_{s\sim\pi^T}\left[u_i(s_i', s_{-i}) - u_i(s_i, s_{-i}) \mid s_i\right] \leq \tilde{\epsilon}$$

- This translates to:

$$\sum_{s_{-i}} \frac{\pi^T(s_i, s_{-i})}{\Pr(s_i)} \cdot \left(u_i(s_i', s_{-i}) - u_i(s_i, s_{-i})\right) \leq \tilde{\epsilon}$$

- We can get this version with $\tilde{\epsilon} = \epsilon \,/\, \Pr(s_i)$
- Actions that are played very infrequently have large $\tilde{\epsilon}$ even if they have small ϵ

Recent example research in multi-agent RL using Correlated Equilibrium Techniques

Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

Luke Marris [1 2] Paul Muller [1 3] Marc Lanctot [1] Karl Tuyls [1] Thore Graepel [1 2]

Abstract

Two-player, constant-sum games are well studied in the literature, but there has been limited progress outside of this setting. We propose Joint Policy-Space Response Oracles (JPSRO), an algorithm for training agents in n-player, general-sum extensive form games, which provably converges to an equilibrium. We further suggest correlated equilibria (CE) as promising meta-solvers, and propose a novel solution concept Maximum Gini Correlated Equilibrium (MGCE), a principled and computationally efficient family of solutions for solving the correlated equilibrium selection problem. We conduct several experiments using CE meta-solvers for JPSRO and demonstrate convergence on n-player, general-sum games.

1. Introduction

Recent success in tackling two-player, constant-sum games (Silver et al., 2016; Vinyals et al., 2019) has outpaced progress in n-player, general-sum games despite a lot of interest (Jaderberg et al., 2019; OpenAI et al., 2019; Brown & Sandholm, 2019; Lockhart et al., 2020; Gray et al., 2020; Anthony et al., 2020). One reason is because Nash equilibrium (NE) (Nash, 1951) is tractable and interchangeable in the two-player, constant-sum setting but becomes intractable (Daskalakis et al., 2009) and potentially non-interchangeable[1] in n-player and general-sum settings. The problem of selecting from multiple solutions is known as the equilibrium selection problem (Goldberg et al., 2013; Avis et al., 2010; Harsanyi & Selten, 1988).[2]

Outside of normal form (NF) games, this problem setting arises in multi-agent training when dealing with empirical games (also called meta-games), where a game payoff tensor is populated with expected outcomes between agents playing an extensive form (EF) game, for example the StarCraft League (Vinyals et al., 2019) and Policy-Space Response Oracles (PSRO) (Lanctot et al., 2017), a recent variant of which reached state-of-the-art results in Stratego Barrage (McAleer et al., 2020).

In this work we propose using correlated equilibrium (CE) (Aumann, 1974) and coarse correlated equilibrium (CCE) as a suitable target equilibrium space for n-player, general-sum games[3]. The (C)CE solution concept has two main benefits over NE; firstly, it provides a mechanism for players to correlate their actions to arrive at mutually higher payoffs and secondly, it is computationally tractable to compute solutions for n-player, general-sum games (Daskalakis et al., 2009). We provide a tractable approach to select from the space of (C)CEs (MG), and a novel training framework that converges to this solution (JPSRO). The result is a set of tools for theoretically solving any complete information[4] multi-agent problem. These tools are amenable to scaling approaches; including utilizing reinforcement learning, function approximation, and online solution solvers, however we leave this to future work.

In Section 2 we provide background on a) correlated equilibrium (CE), an important generalization of NE, b) coarse correlated equilibrium (CCE) (Moulin & Vial, 1978), a similar solution concept, and c) PSRO, a powerful multi-agent training algorithm. In Section 3 we propose novel solution concepts called Maximum Gini (Coarse) Correlated Equilibrium (MG(C)CE) and in Section 4 we thoroughly explore its properties including tractability, scalability, invariance, and

[1]DeepMind [2]University College London [3]Université Gustave Eiffel. Correspondence to: Luke Marris <marris@google.com>.

Proceedings of the 38th International Conference on Machine

[2]The equilibrium selection problem is subtle and can have various interpretations. We describe it fully in Section 4.1 based