The Keychain Problem: Minimizing the Opportunity Cost of Uncertainty

EC'26, Rome, Italy

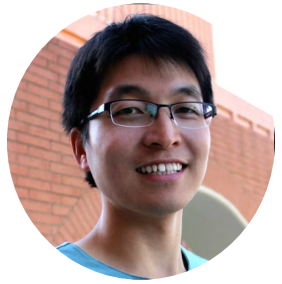
Ramiro Deo-Campo Vuong
Cornell University

Robert Kleinberg
Cornell University

Aditya Prasad
University of Chicago

Eric Xiao
National Taiwan University

Haifeng Xu
University of Chicago



This Paper in a Nutshell

The Keychain Problem

This Paper in a Nutshell

The Keychain Problem

a novel yet basic model about *constrained exploration*

This Paper in a Nutshell

The Keychain Problem

a novel yet basic model about *constrained exploration*

This Paper in a Nutshell

The Keychain Problem

a novel yet basic model about *constrained exploration*

This Paper in a Nutshell

The Keychain Problem

a novel yet basic model about *constrained exploration*

This Paper in a Nutshell

The Keychain Problem

a novel yet basic model about *constrained exploration*

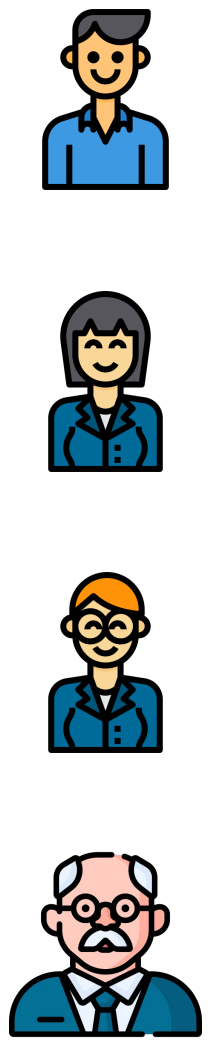
The Basic Keychain Problem

The Basic Keychain Problem

- Testing candidate treatments for a new disease

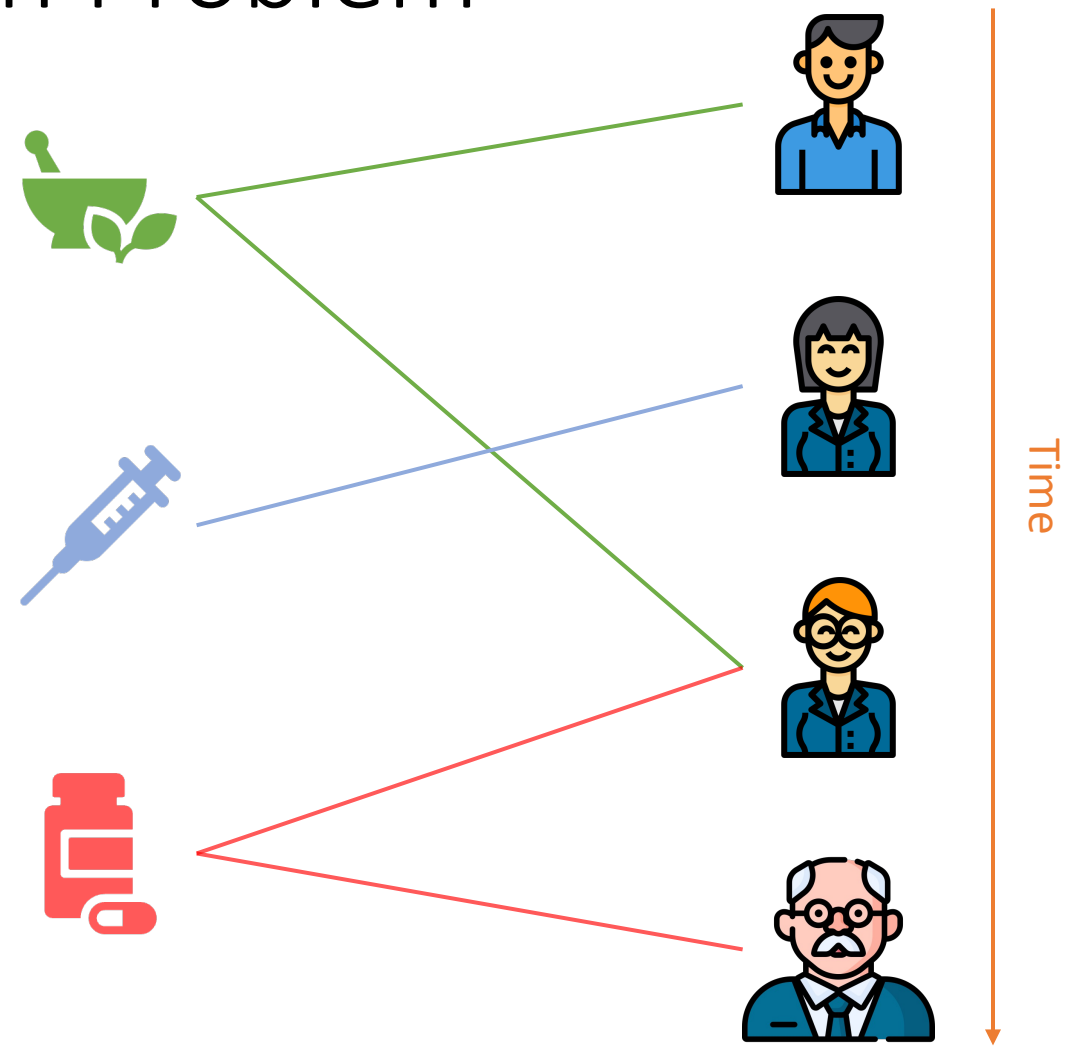
The Basic Keychain Problem

- Testing candidate treatments for a new disease
- Patients arrive in a known order



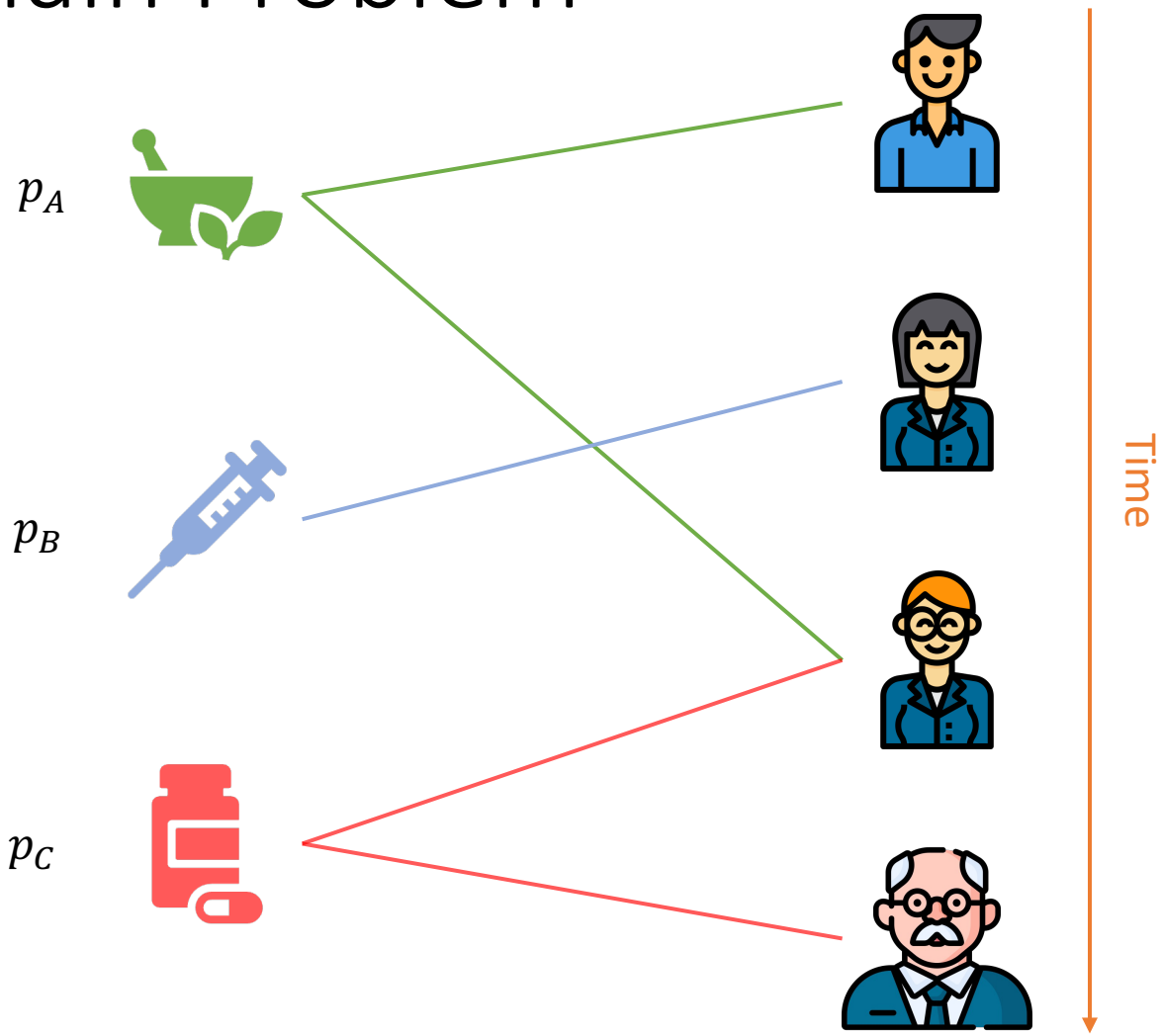
The Basic Keychain Problem

- Testing candidate treatments for a new disease
- Patients arrive in a known order
- Not all treatments can be applied to all patients



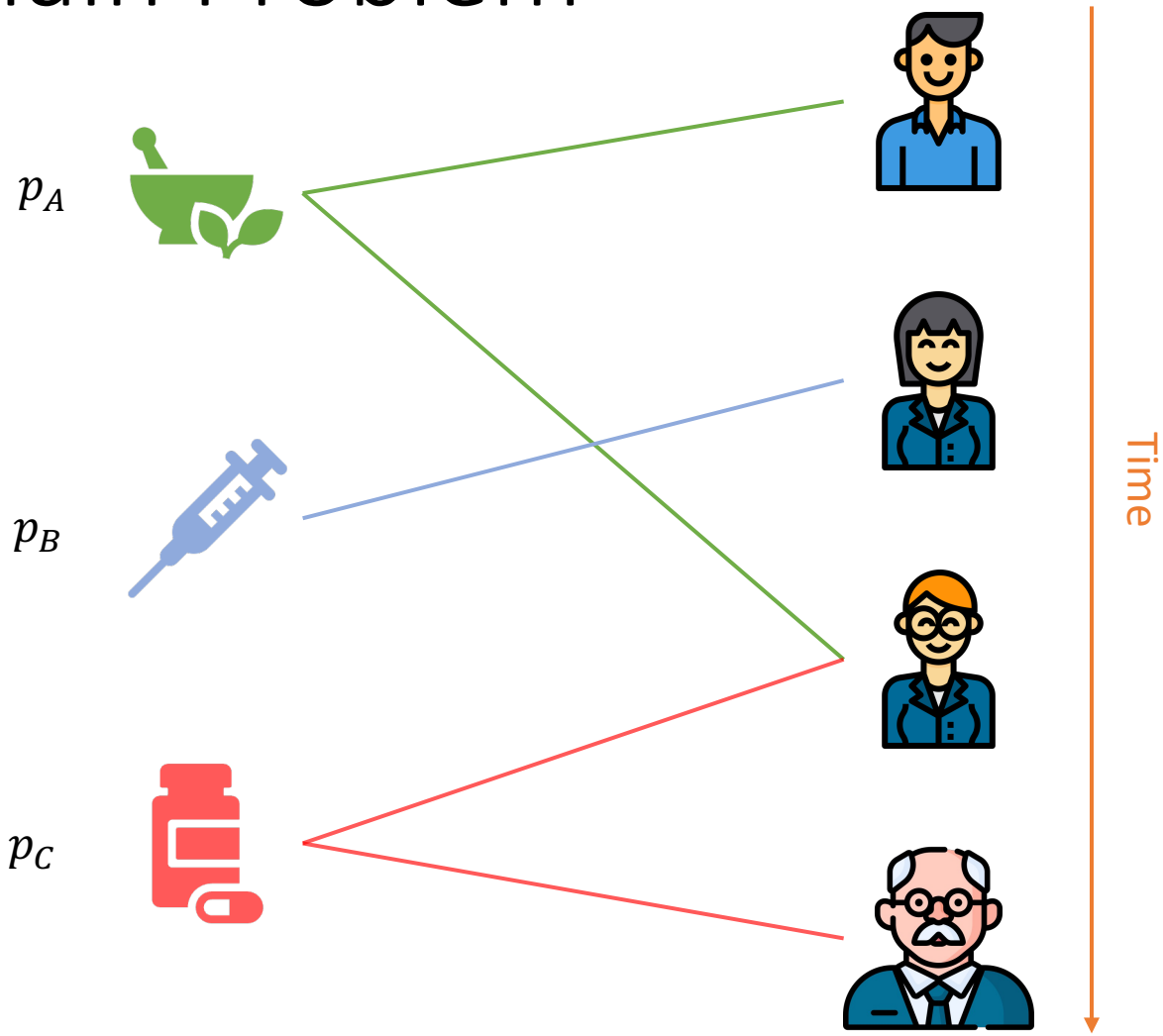
The Basic Keychain Problem

- Testing candidate treatments for a new disease
- Patients arrive in a known order
- Not all treatments can be applied to all patients
- Only one treatment works, prior belief on success of each treatment

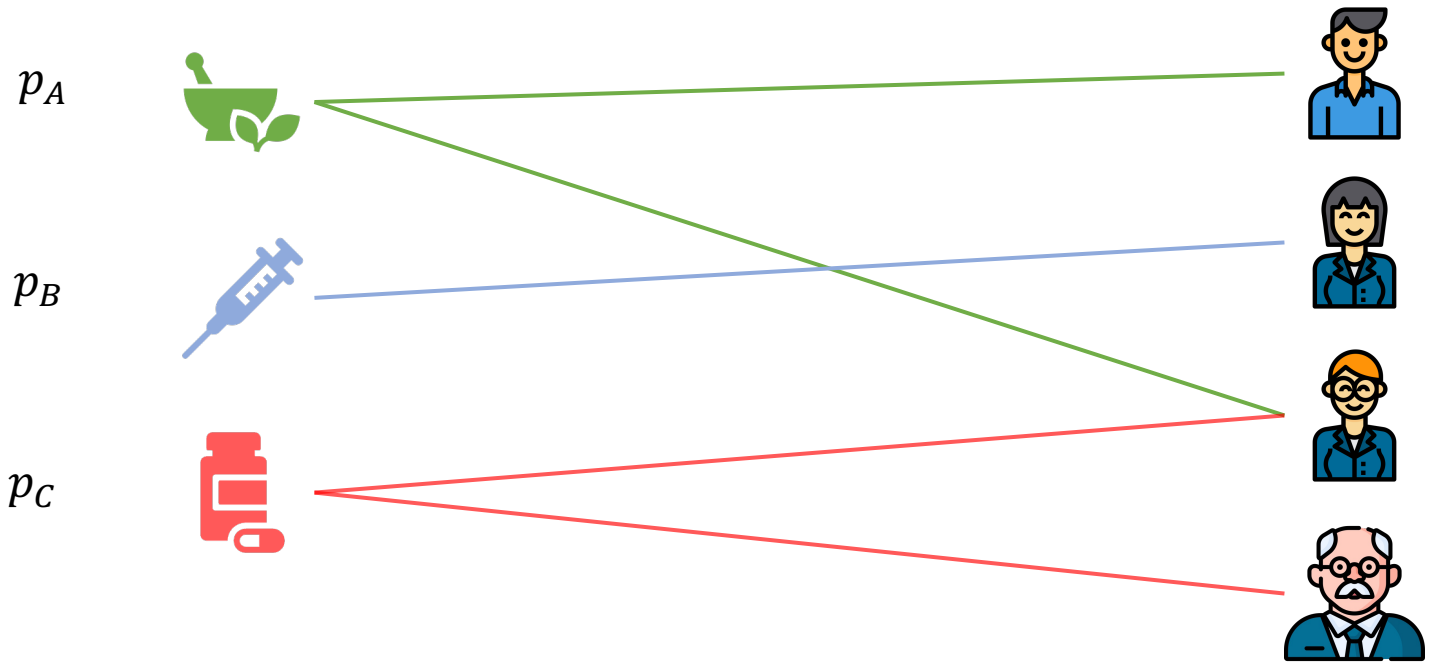


The Basic Keychain Problem

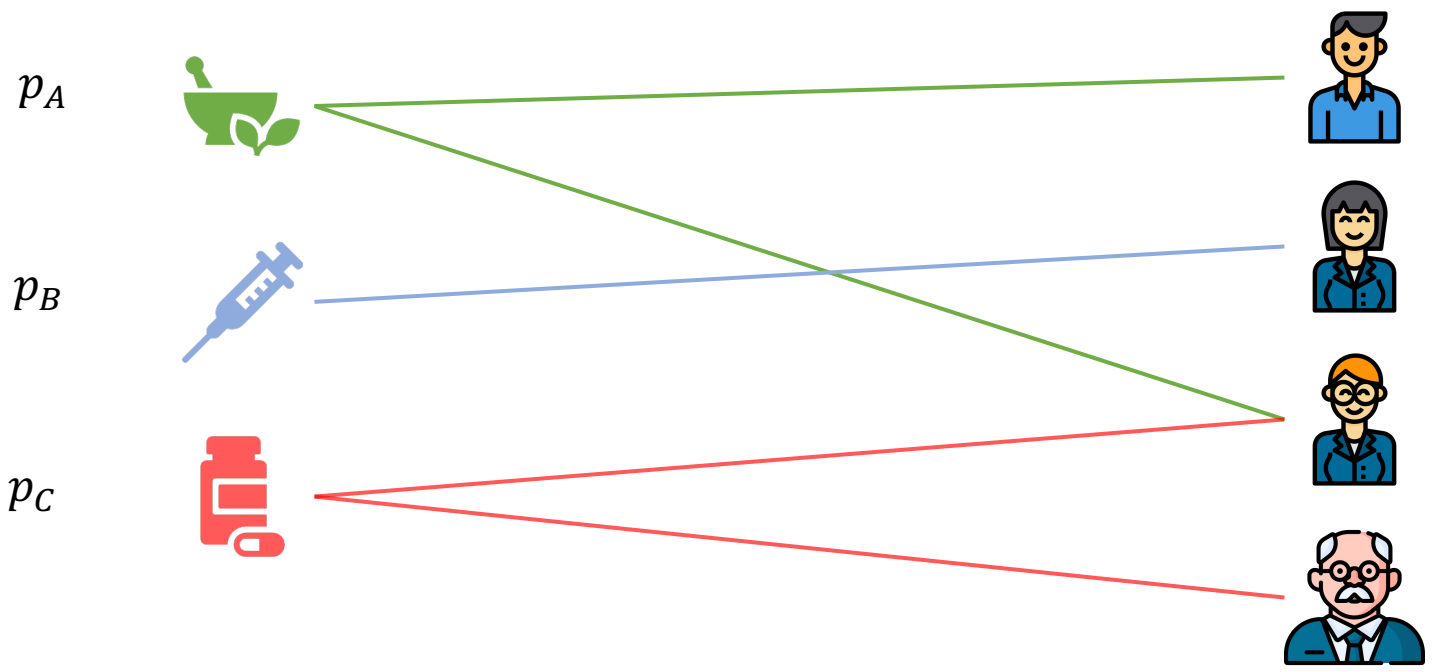
- Testing candidate treatments for a new disease
- Patients arrive in a known order
- Not all treatments can be applied to all patients
- Only one treatment works, prior belief on success of each treatment
- Cure the most patients



The Basic Keychain Problem

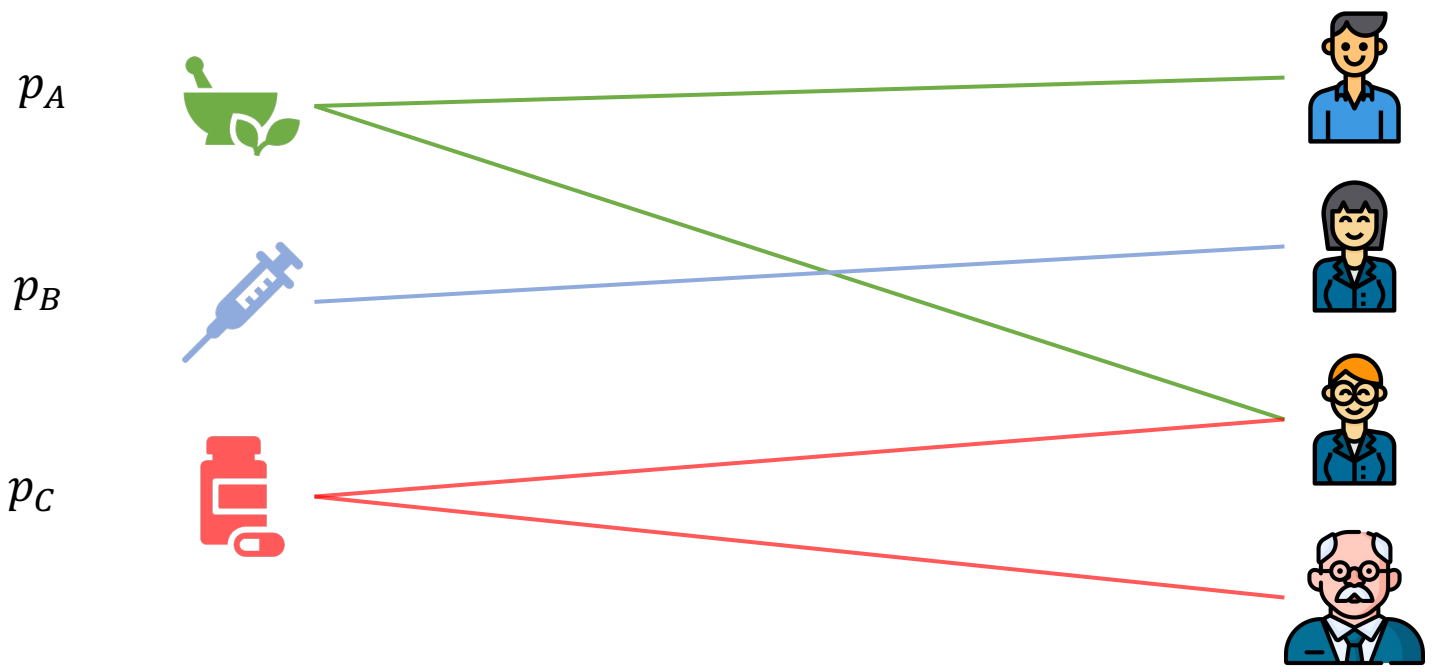


The Basic Keychain Problem



Insight:
Only one correct treatment exists - once it is found, exploit it.

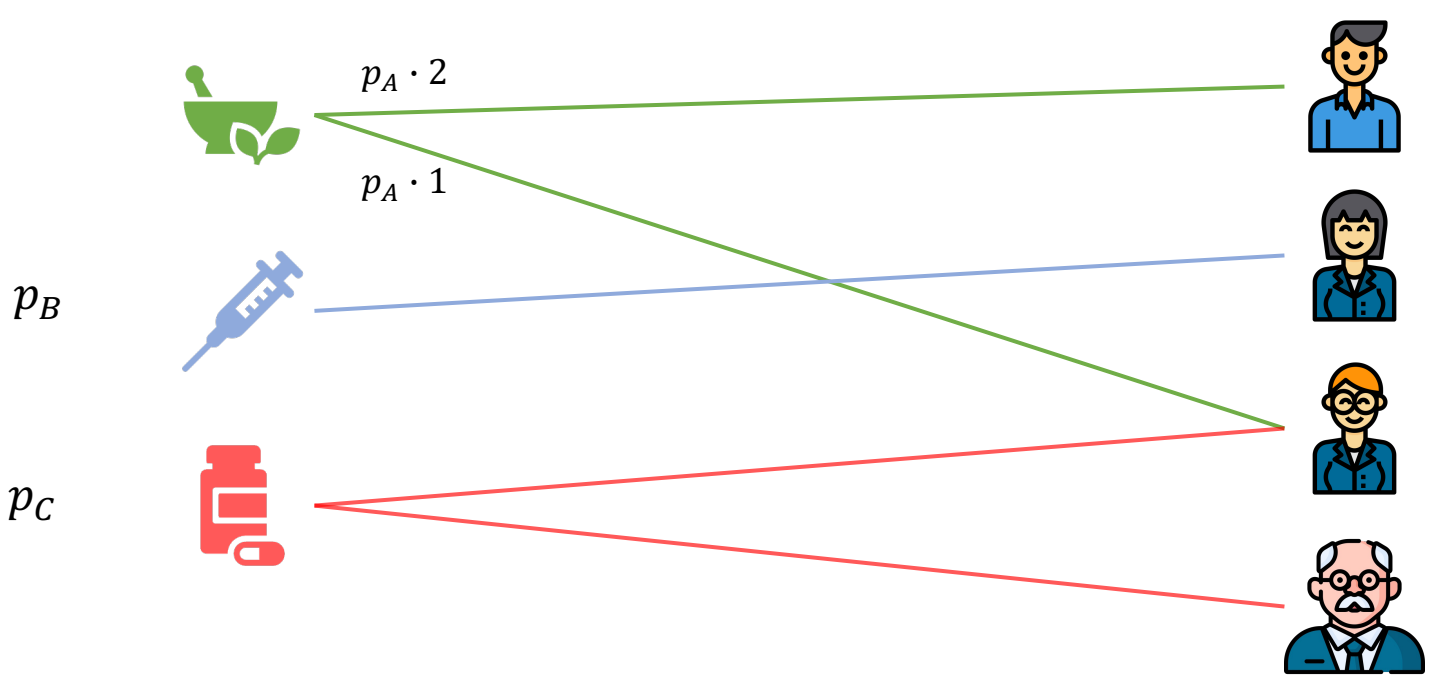
The Basic Keychain Problem



Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
p_k · # future patients compatible with k

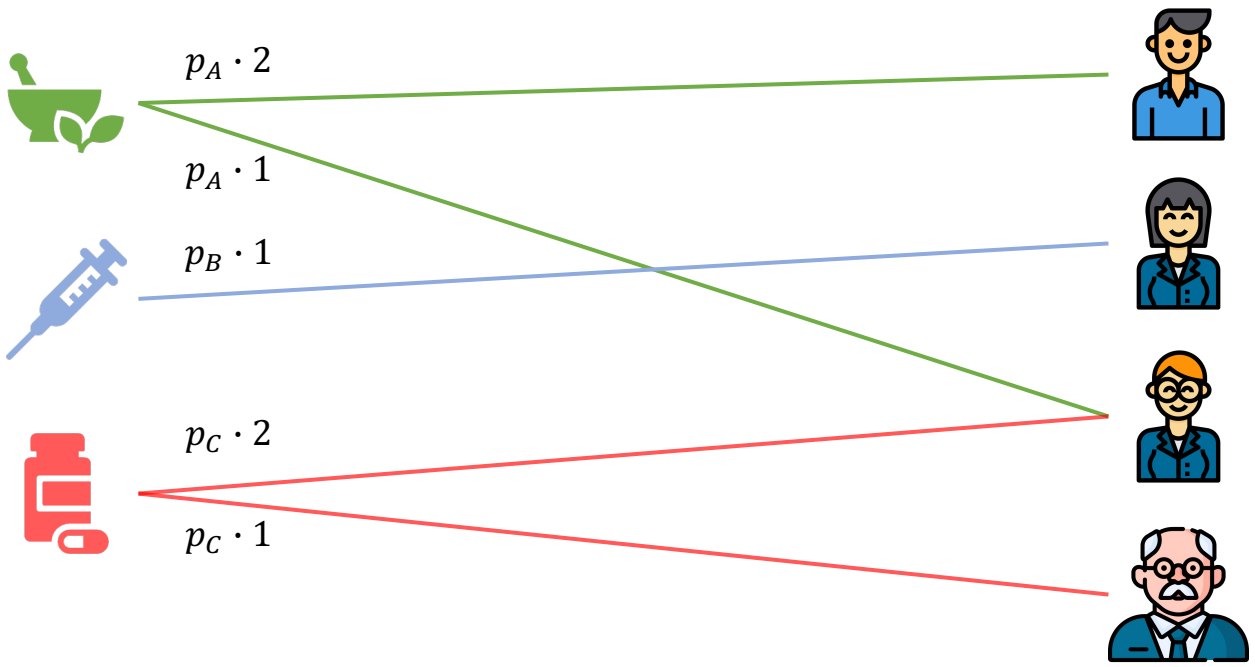
The Basic Keychain Problem



Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
$p_k \cdot$ # future patients compatible with k

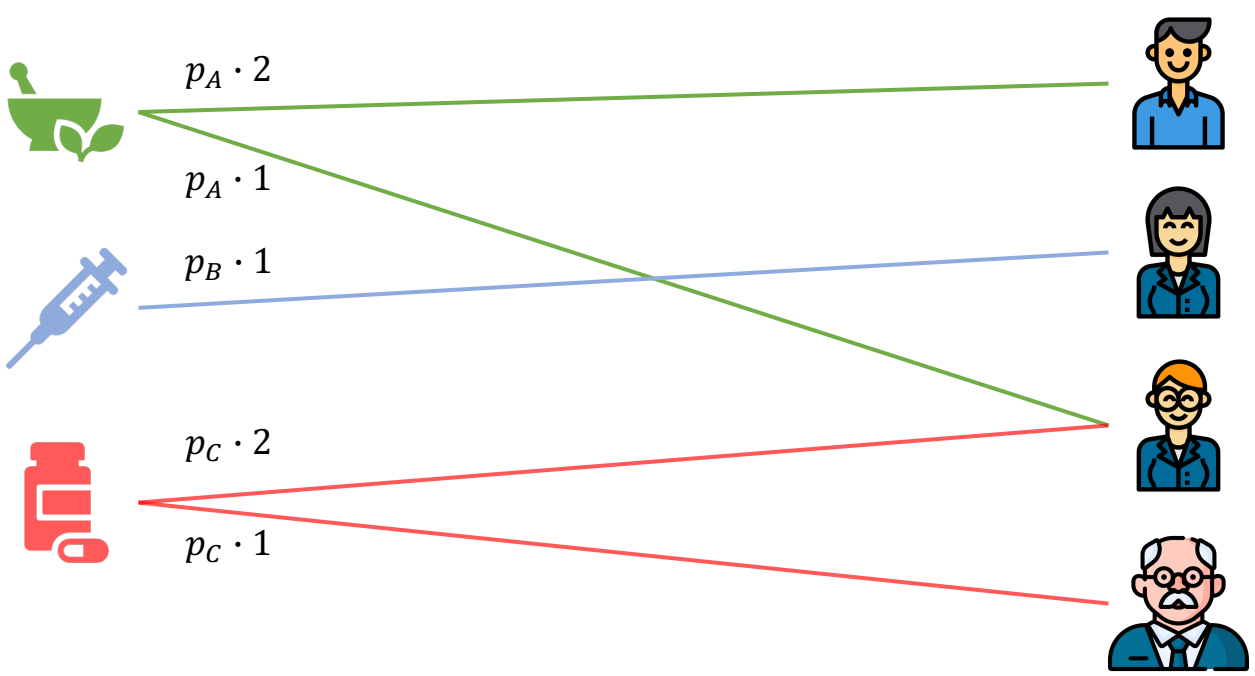
The Basic Keychain Problem



Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
p_k · # future patients compatible with k

The Basic Keychain Problem

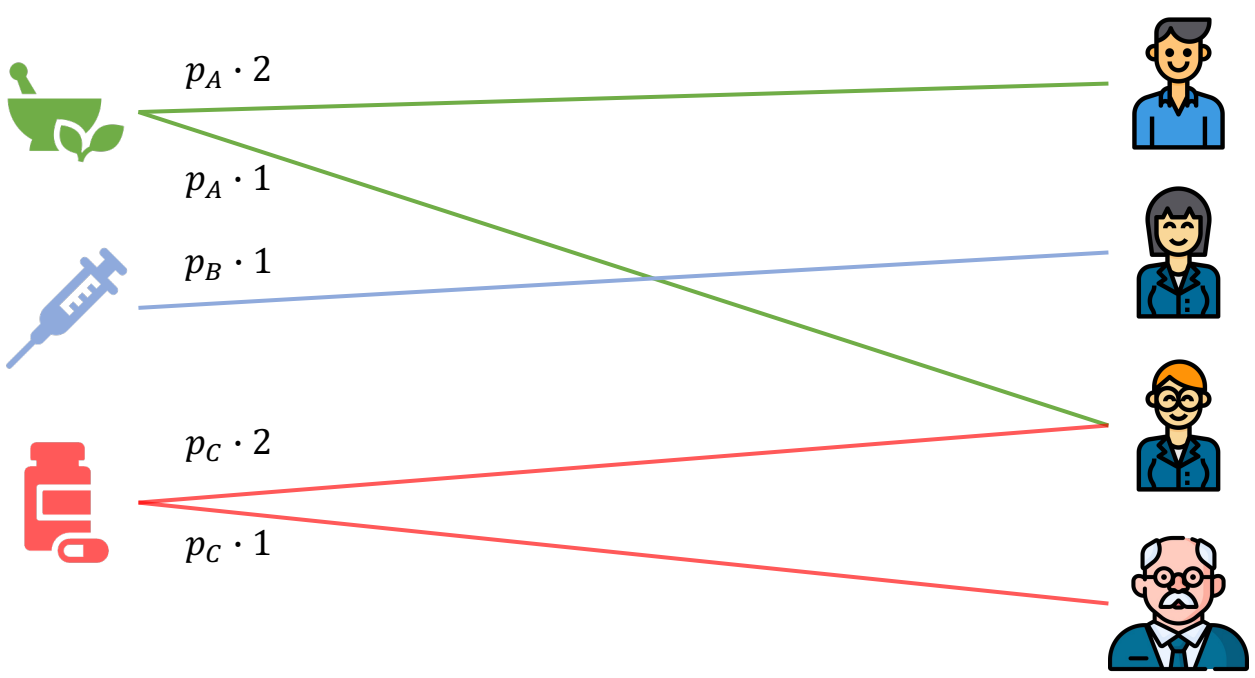


Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
$p_k \cdot$ # future patients compatible with k

Matching - Edge (k, t) is selected:
Treatment k is tried **for the first time** on patient C_t

The Basic Keychain Problem



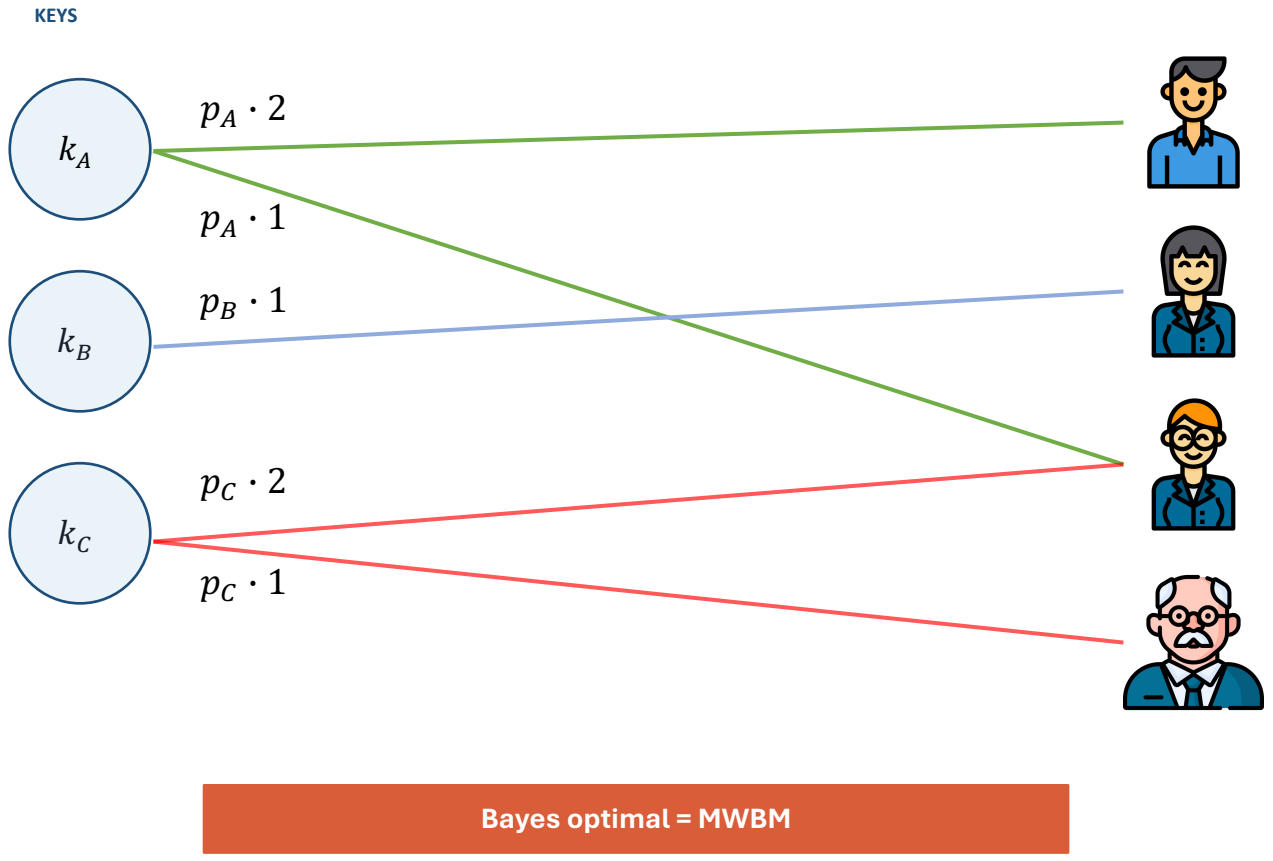
Bayes optimal = MWBM

Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
$p_k \cdot$ # future patients compatible with k

Matching - Edge (k, t) is selected:
Treatment k is tried **for the first time** on patient C_t

The Basic Keychain Problem



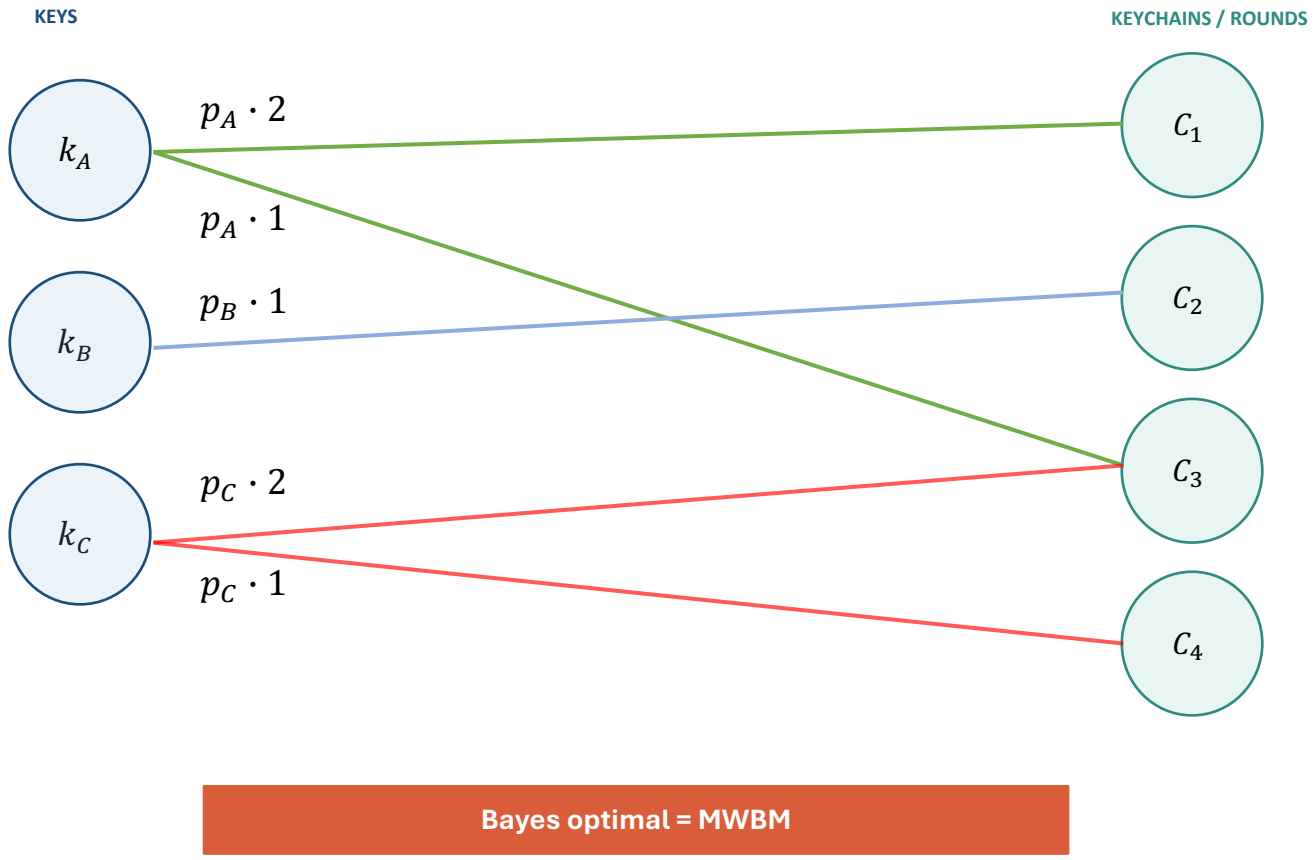
Bayes optimal = MWBM

Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
$p_k \cdot$ # future patients compatible with k

Matching - Edge (k, t) is selected:
Treatment k is tried **for the first time** on patient C_t

The Basic Keychain Problem



Bayes optimal = MWBM

Insight:
Only one correct treatment exists - once it is found, exploit it.

Edge weight – expected future reward:
$p_k \cdot$ # future patients compatible with k

Matching - Edge (k, t) is selected:
Treatment k is tried **for the first time** on patient C_t

Why the Keychain Problem Matters

Why the Keychain Problem Matters

- Connections to well-studied theoretical problems:

Why the Keychain Problem Matters

- Connections to well-studied theoretical problems:
  - Bipartite graph problems

Why the Keychain Problem Matters

- Connections to well-studied theoretical problems:
  - Bipartite graph problems
  - Bandit optimization problems
  - Stochastic Search/Probing

Why the Keychain Problem Matters

- Connections to well-studied theoretical problems:
  - Bipartite graph problems
  - Bandit optimization problems
  - Stochastic Search/Probing

- Viable model for real-world applications:
  - Clinical trials/vaccine testing
  - Finding best hiring signals
  - Red-teaming for network vulnerability testing

Why the Keychain Problem Matters

- Connections to well-studied theoretical problems:
  - Bipartite graph problems
  - Bandit optimization problems
  - Stochastic Search/Probing

- Viable model for real-world applications:
  - Clinical trials/vaccine testing
  - Finding best hiring signals
  - Red-teaming for network vulnerability testing

Variants

Question: What if the order is fixed and known?

Setting: Basic Keychain Problem

Question: What if the order is drawn from a known distribution?

Setting: Probabilistic Scenarios

Question: What if we can select the order of keychains?

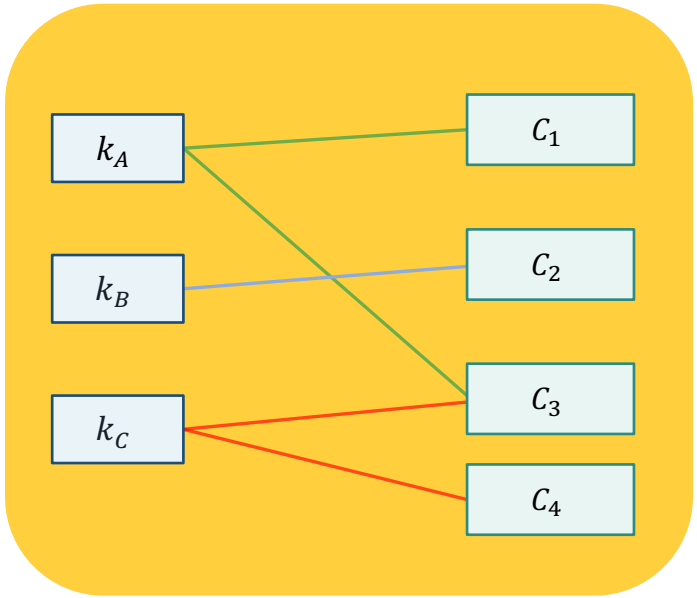
Setting: Order Selection

Fixed-Order Probabilistic Scenarios

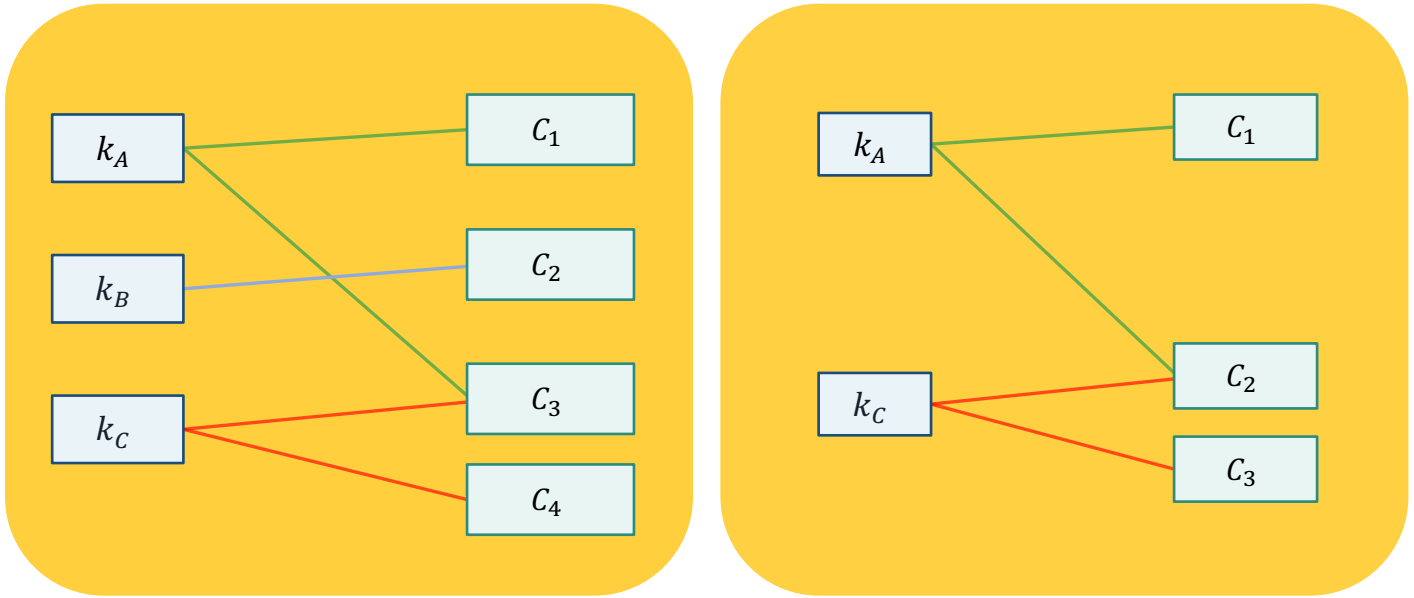
Question: What if the order is drawn from a known distribution?

Setting: Probabilistic Scenarios

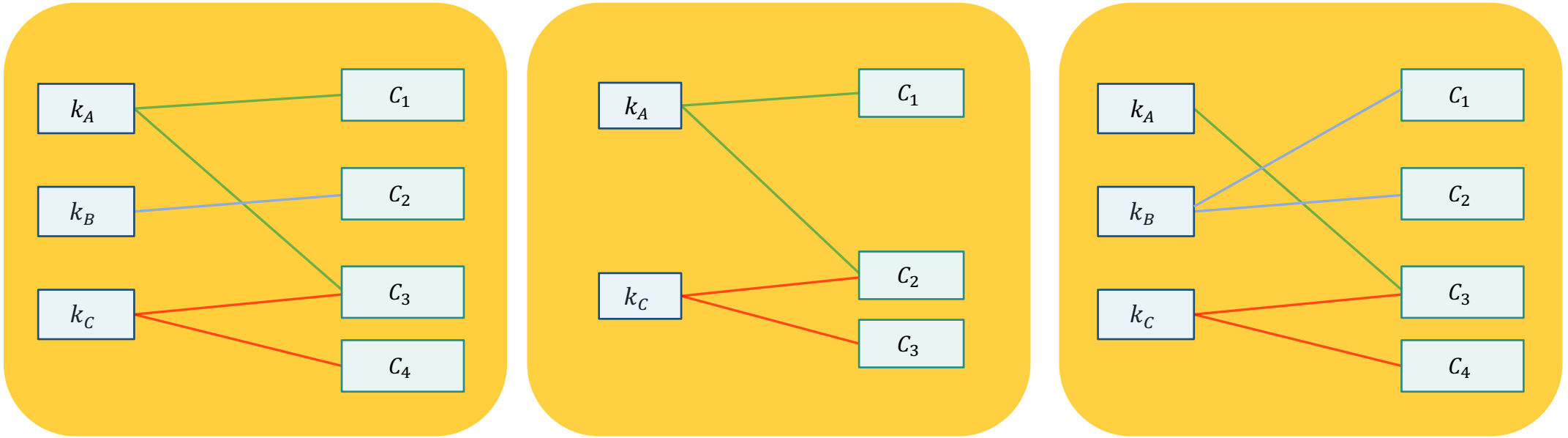
Fixed-Order Probabilistic Scenarios



Fixed-Order Probabilistic Scenarios

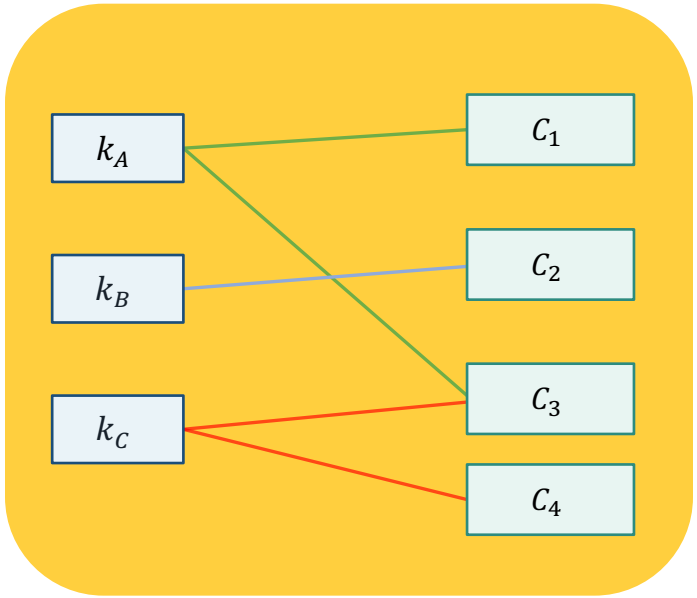


Fixed-Order Probabilistic Scenarios

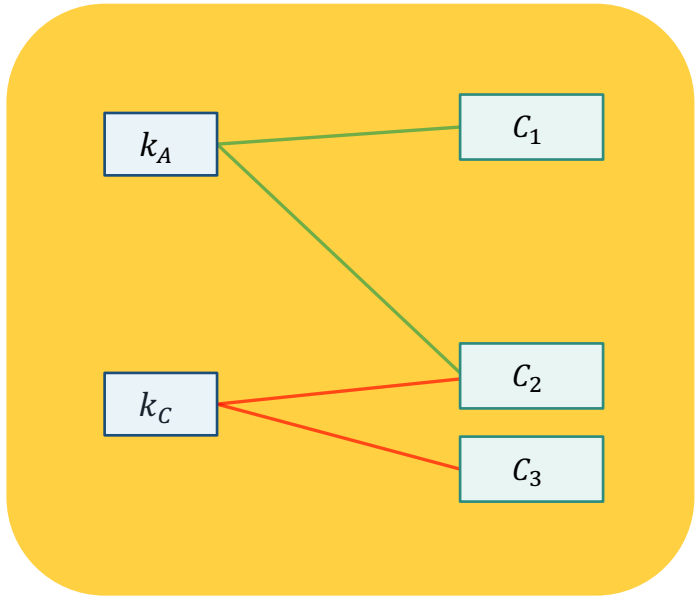


Fixed-Order Probabilistic Scenarios

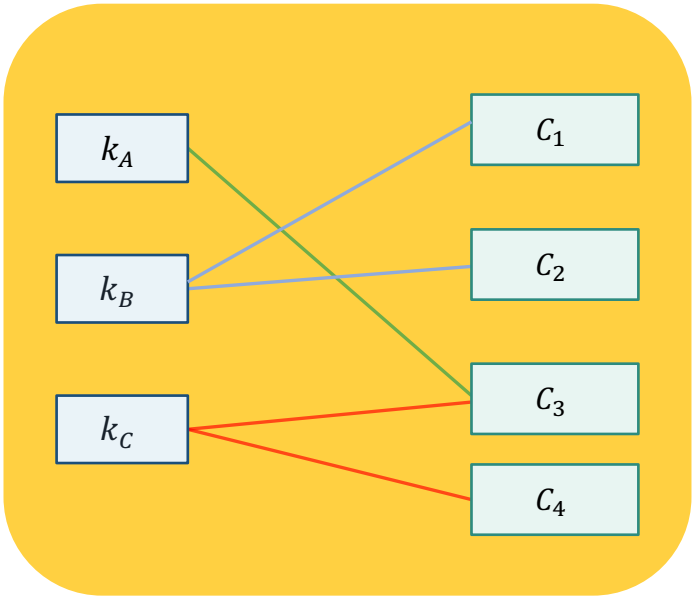
Scenario 1: $p_1 = 0.25$



Scenario 2: $p_2 = 0.5$

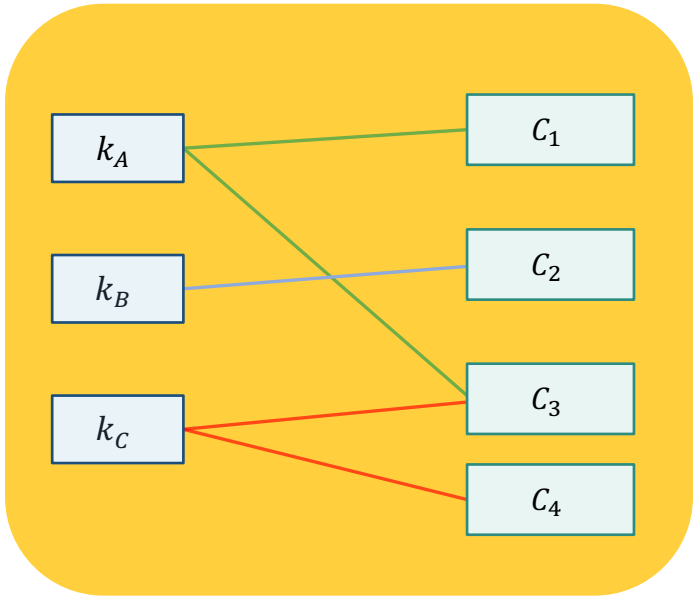


Scenario 3: $p_3 = 0.25$

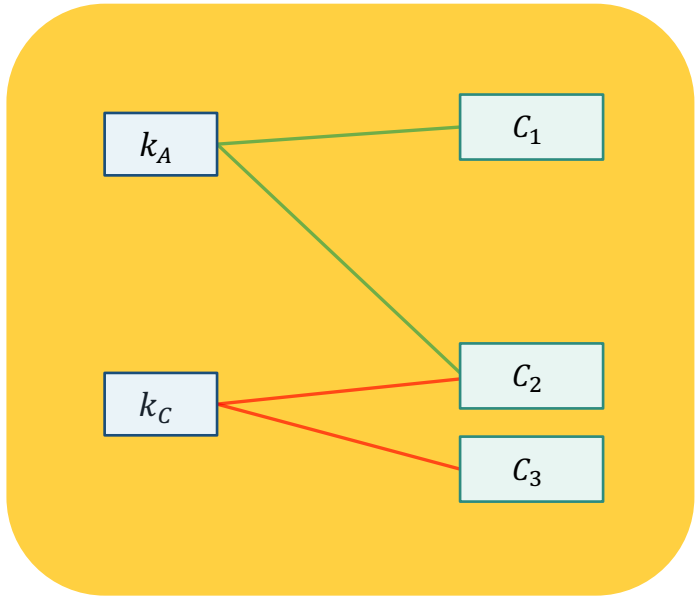


Fixed-Order Probabilistic Scenarios

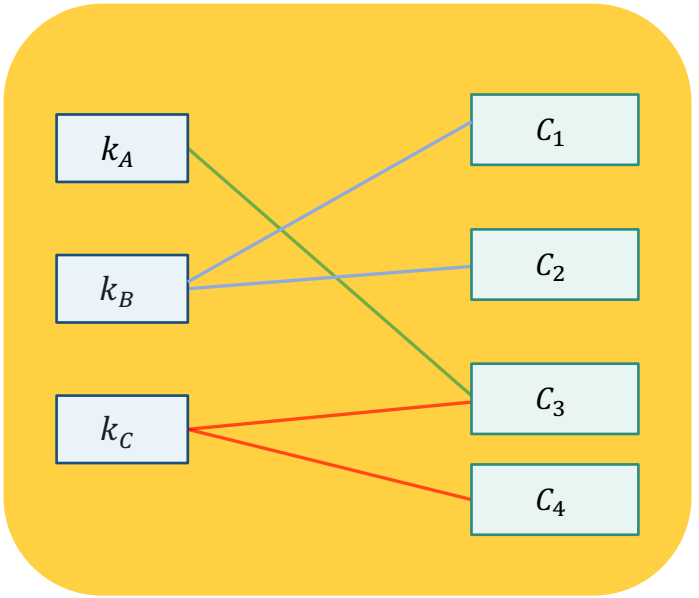
Scenario 1: $p_1 = 0.25$



Scenario 2: $p_2 = 0.5$



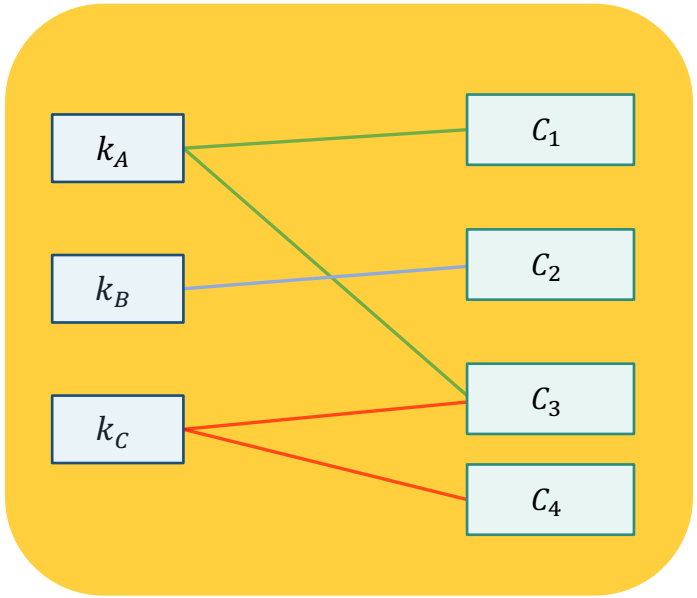
Scenario 3: $p_3 = 0.25$



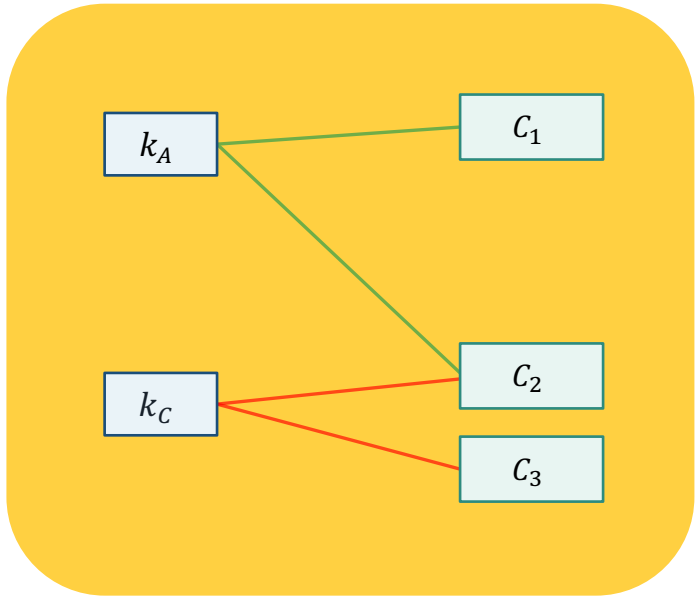
After observing $C_1 = \{k_A\}$, we can safely rule out scenario 3, update probabilities, play accordingly.

Fixed-Order Probabilistic Scenarios

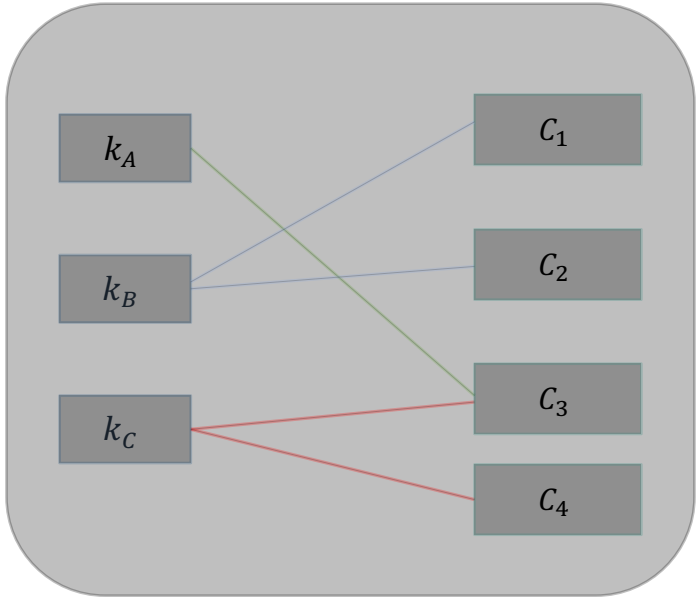
Scenario 1: $p_1 = 0.25$



Scenario 2: $p_2 = 0.5$



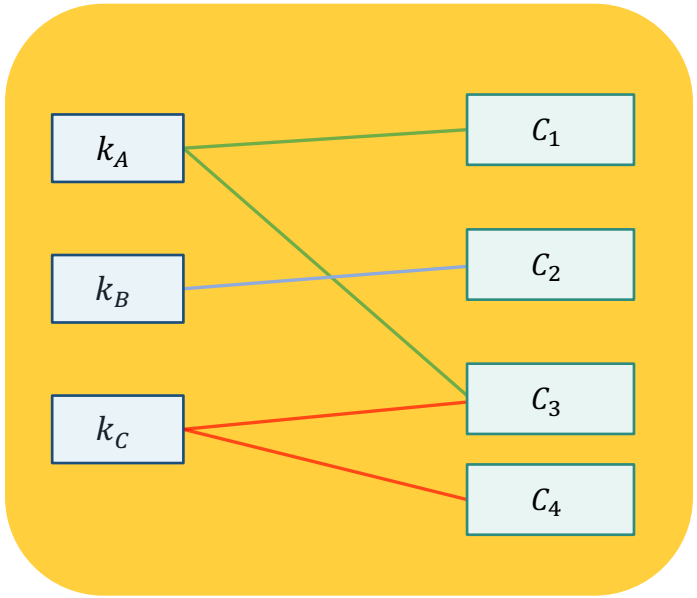
Scenario 3: $p_3 = 0.25$



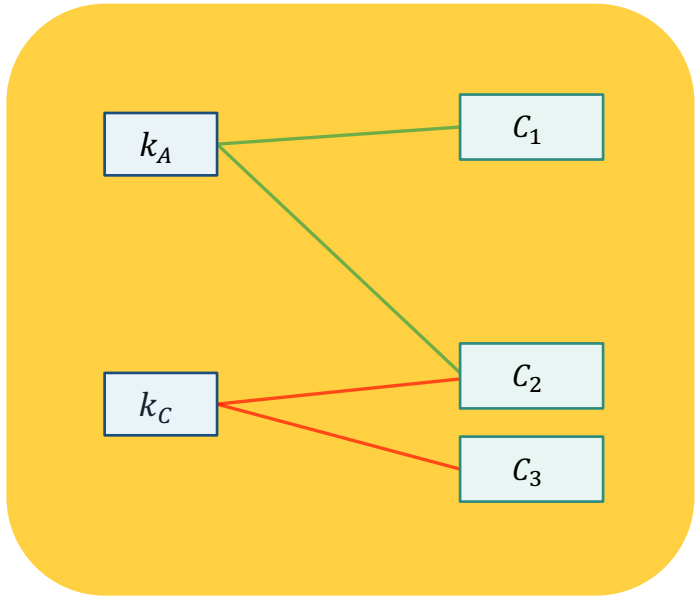
After observing $C_1 = \{k_A\}$, we can safely rule out scenario 3, update probabilities, play accordingly.

Fixed-Order Probabilistic Scenarios

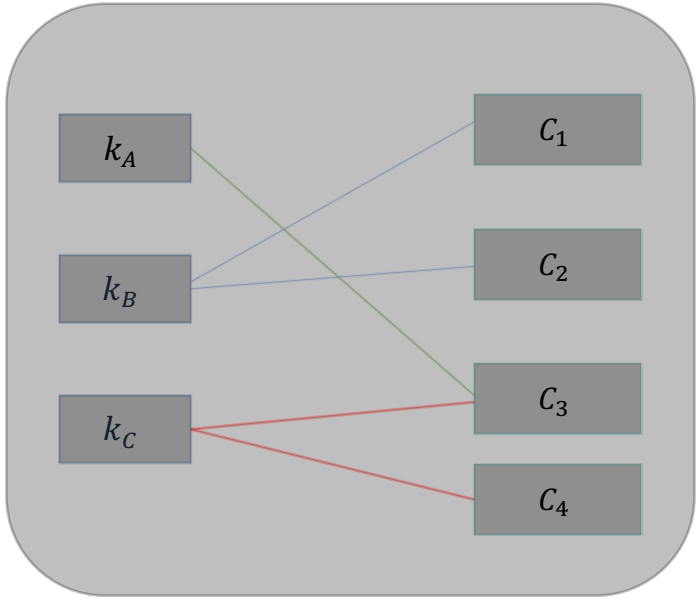
Scenario 1: $p_1 = 0.33$



Scenario 2: $p_2 = 0.66$



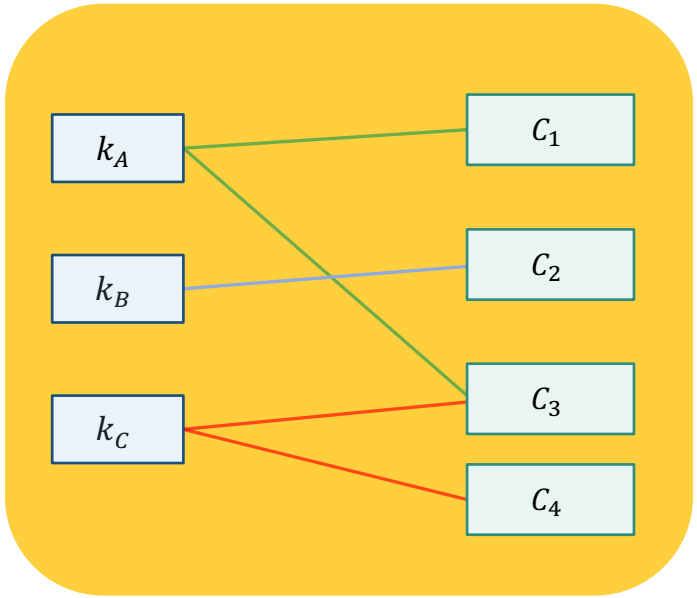
Scenario 3: $p_3 = 0.25$



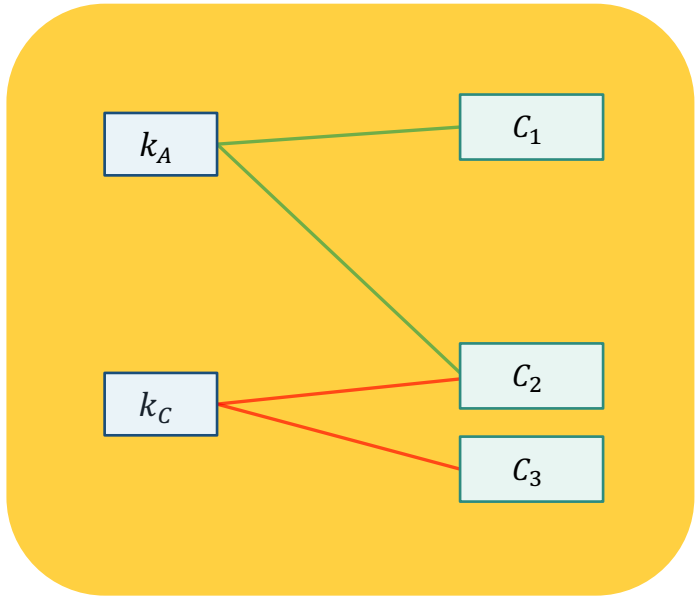
After observing $C_1 = \{k_A\}$, we can safely rule out scenario 3, update probabilities, play accordingly.

Fixed-Order Probabilistic Scenarios

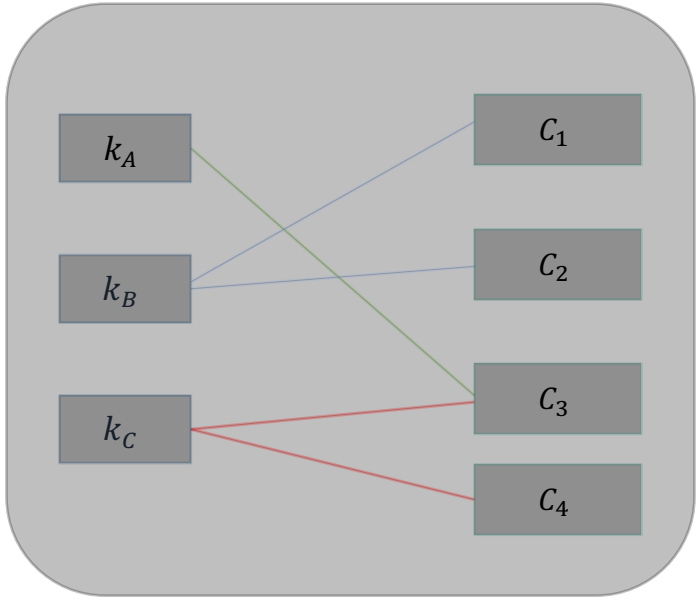
Scenario 1: $p_1 = 0.33$



Scenario 2: $p_2 = 0.66$



Scenario 3: $p_3 = 0.25$



After observing $C_1 = \{k_A\}$, we can safely rule out scenario 3, update probabilities, play accordingly.

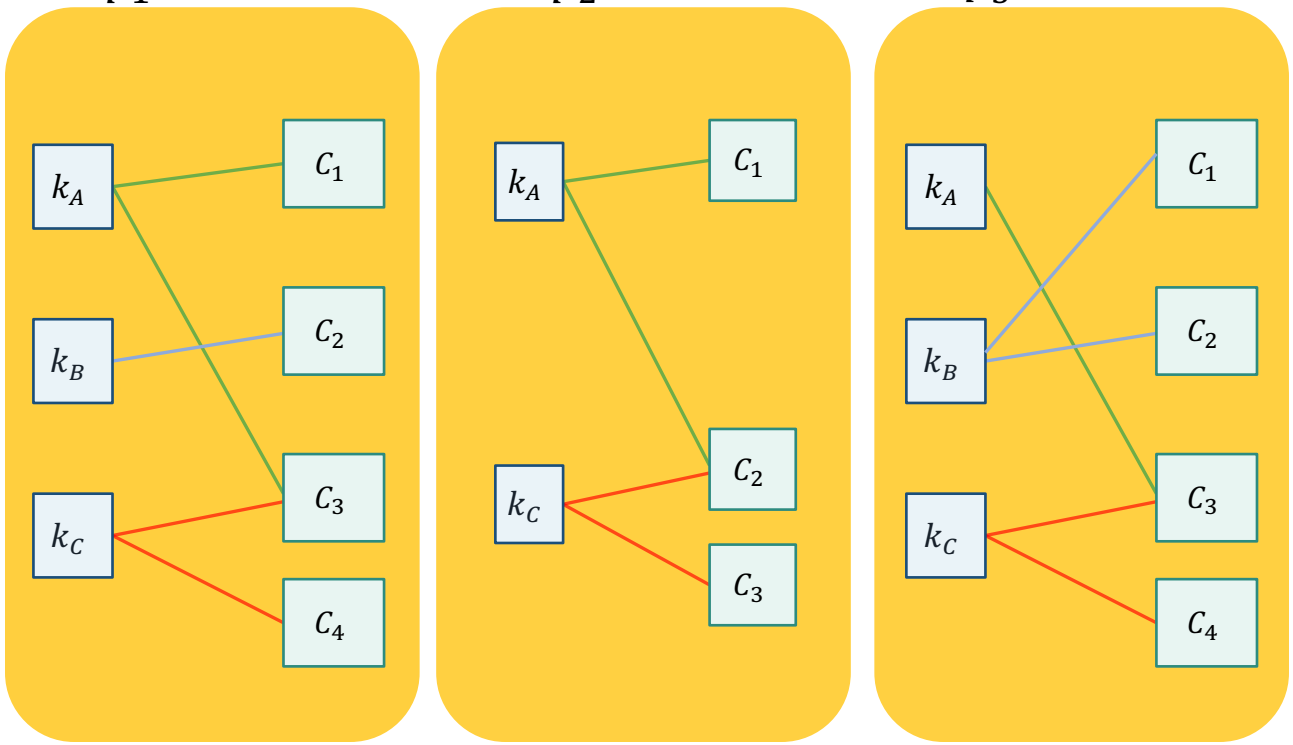
Overarching question: What policy maximizes reward?

Policies Depend on Observation Histories

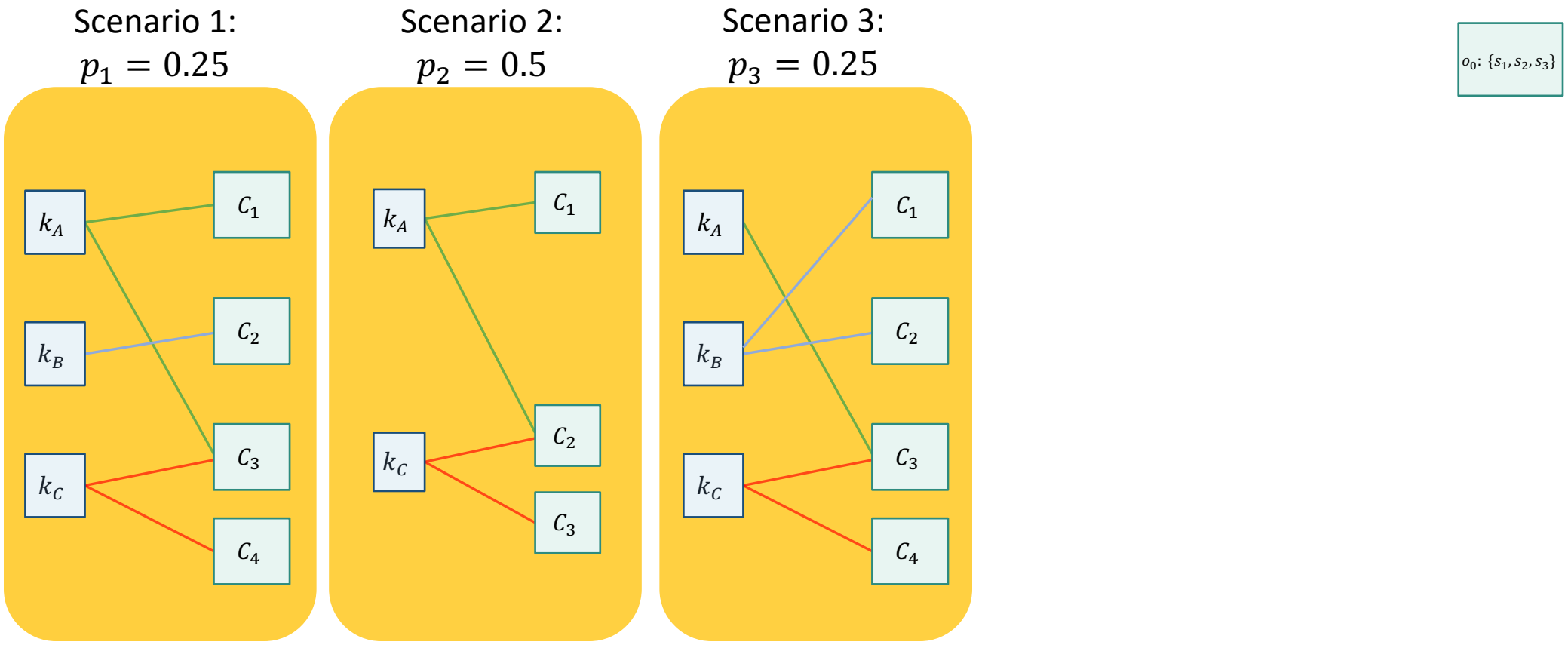
Scenario 1:
$p_1 = 0.25$

Scenario 2:
$p_2 = 0.5$

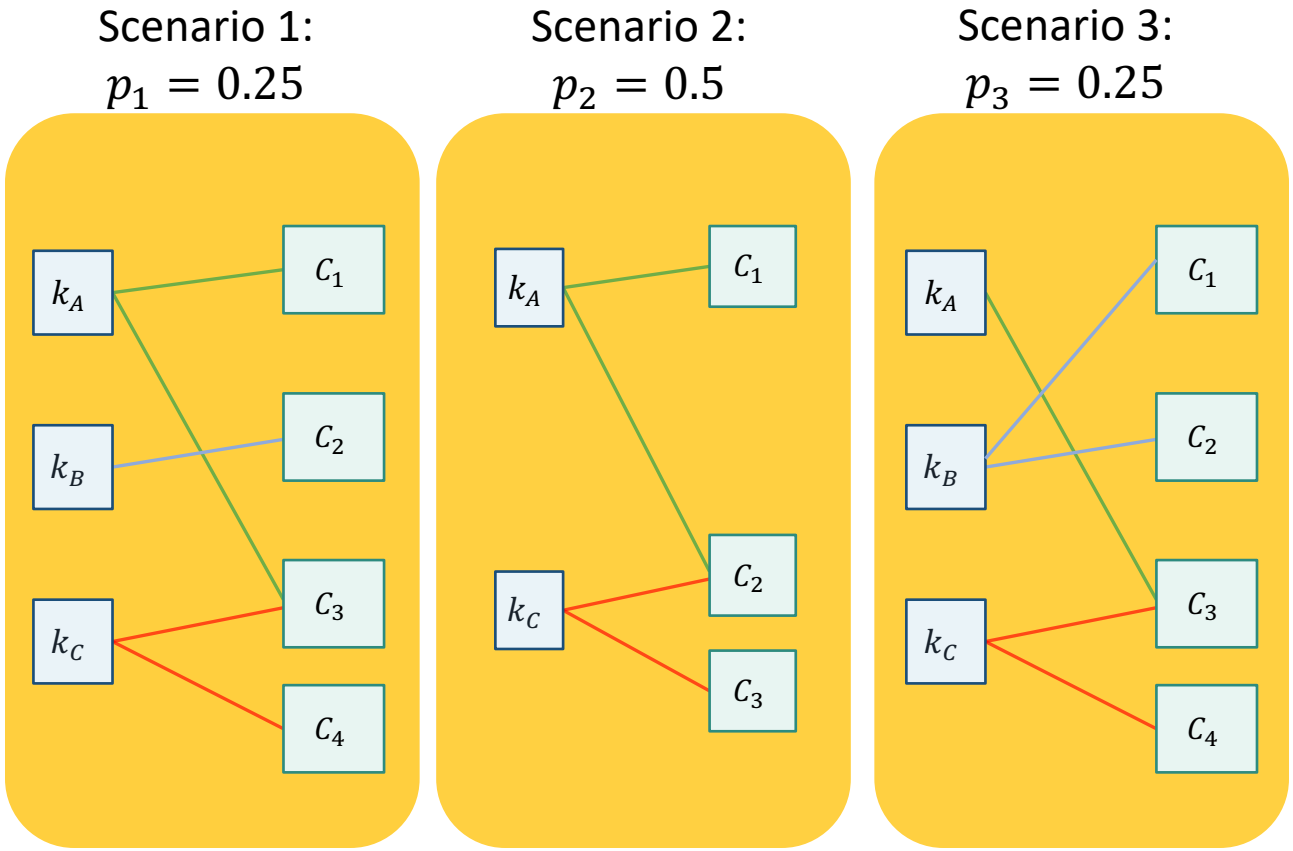
Scenario 3:
$p_3 = 0.25$

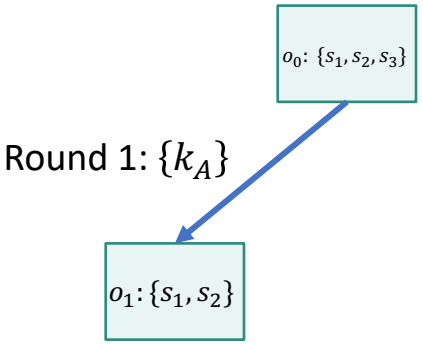


Policies Depend on Observation Histories

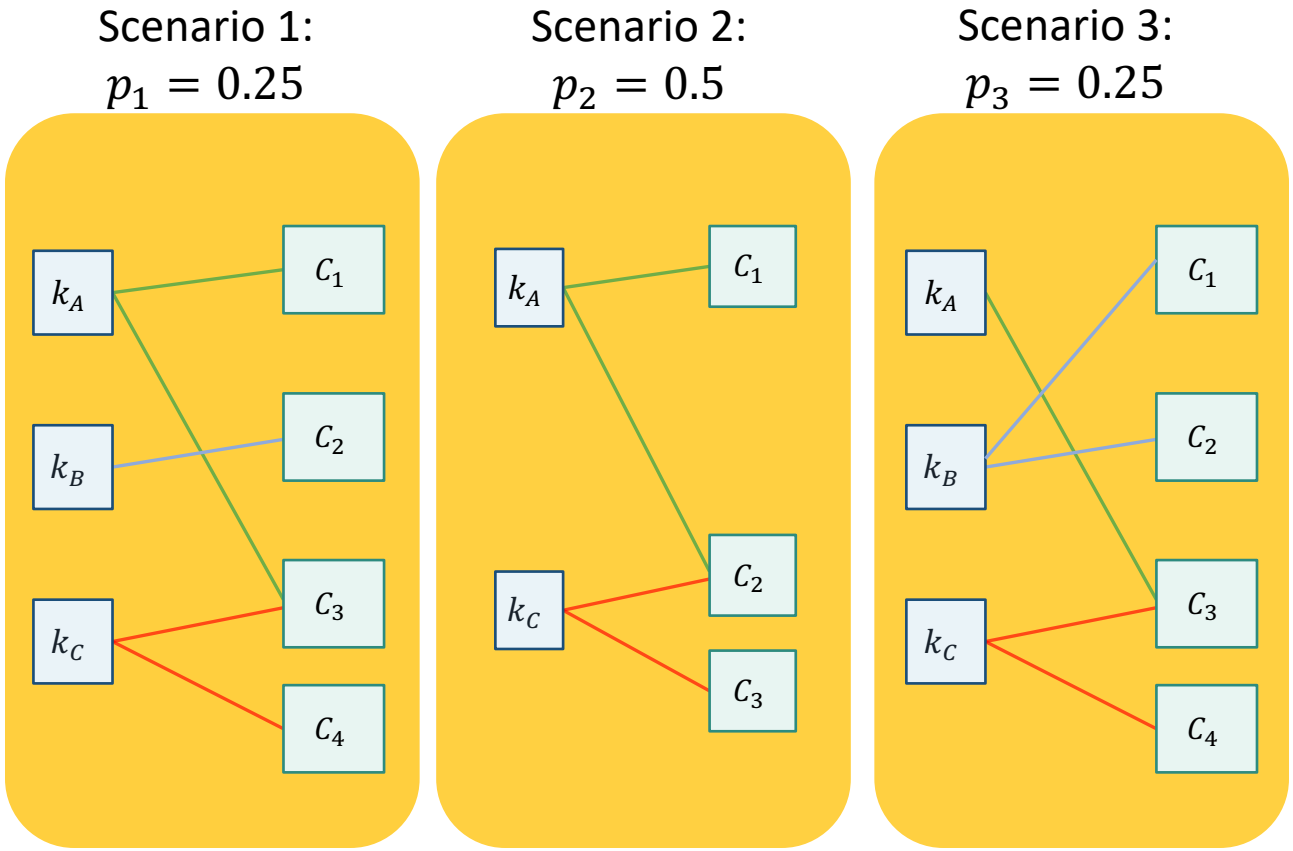


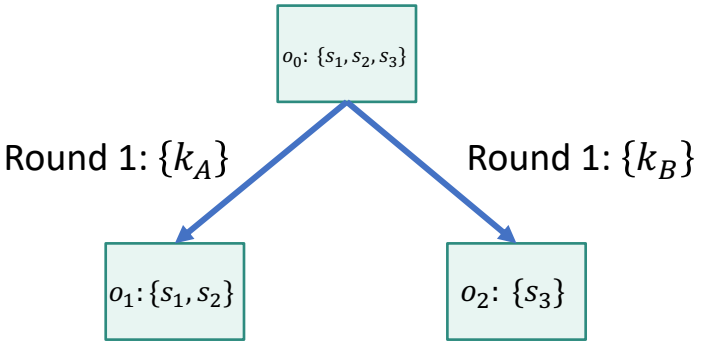
Policies Depend on Observation Histories



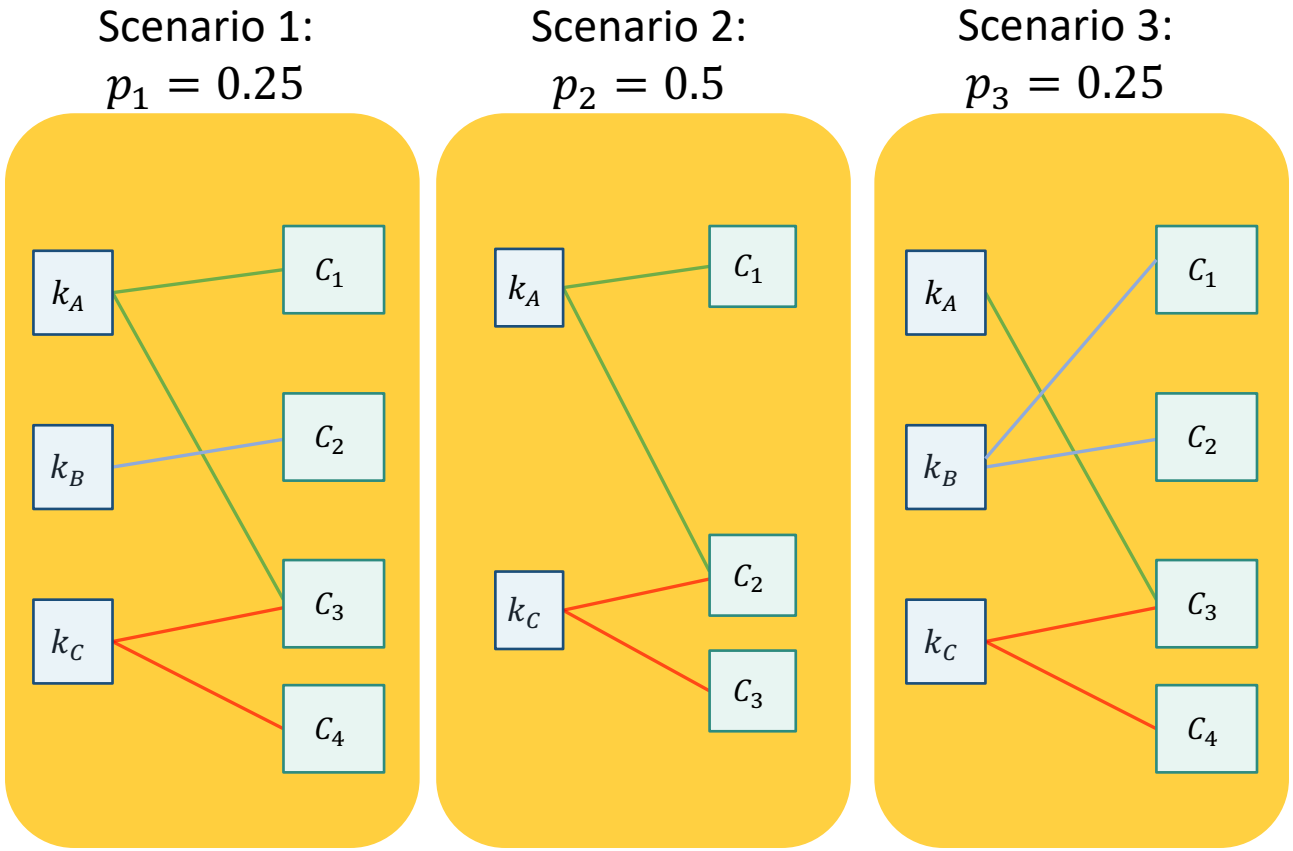


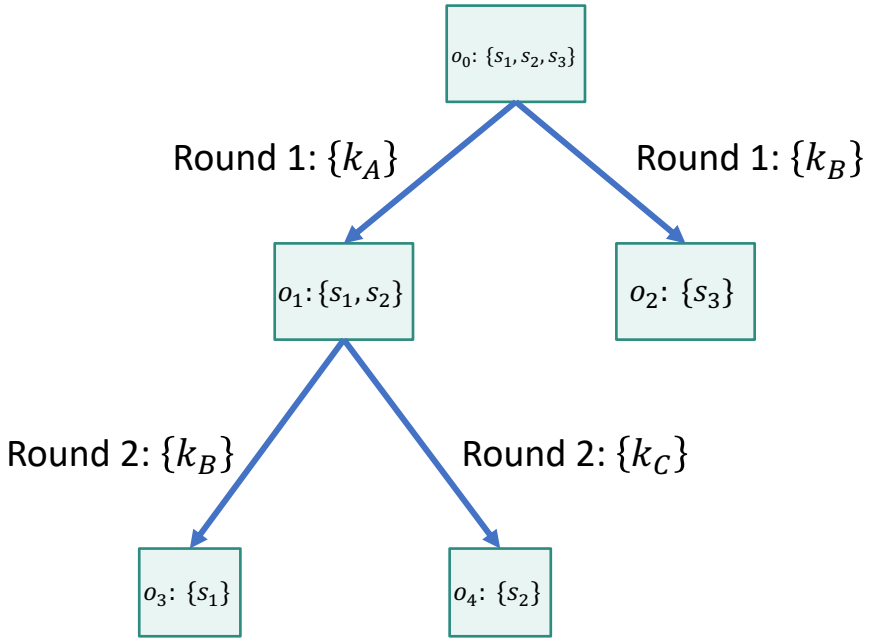
Policies Depend on Observation Histories



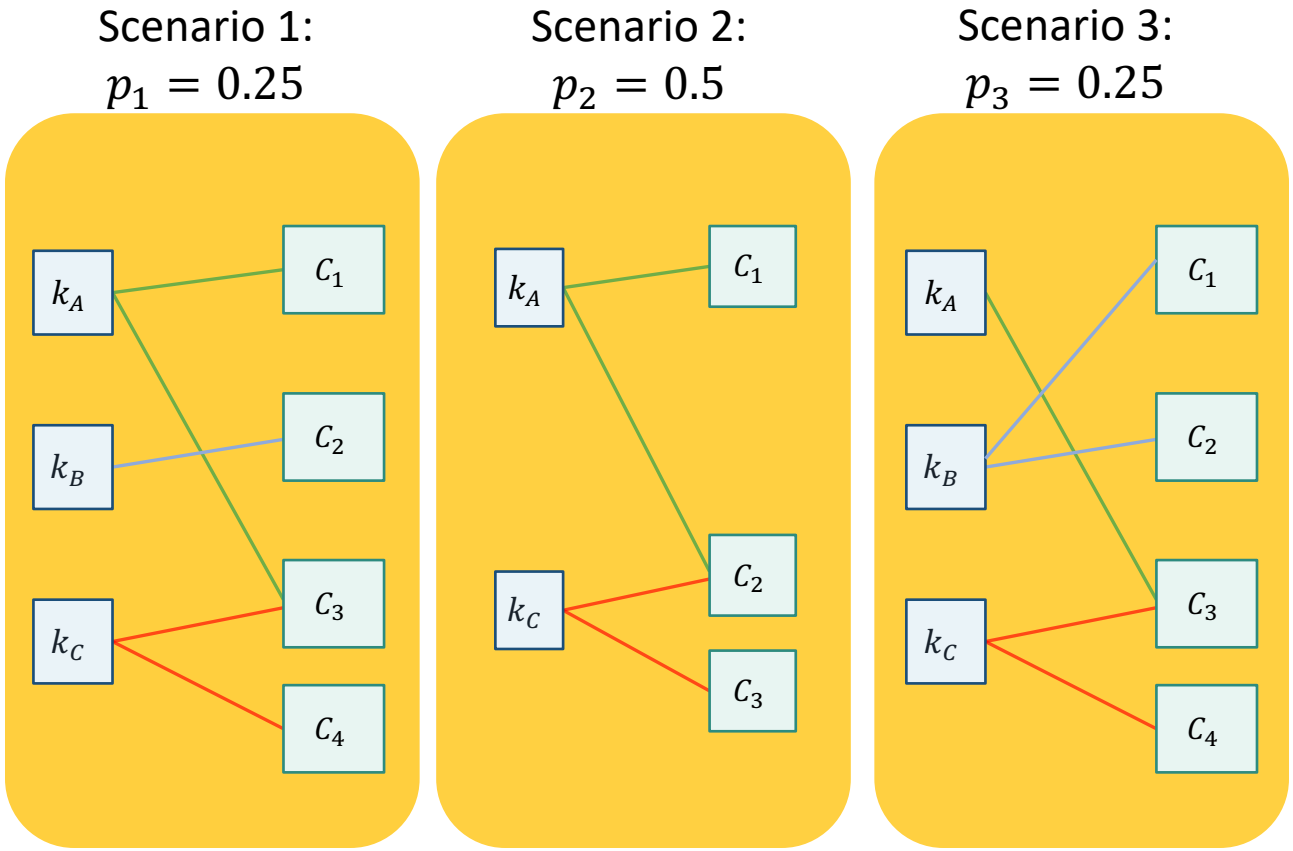


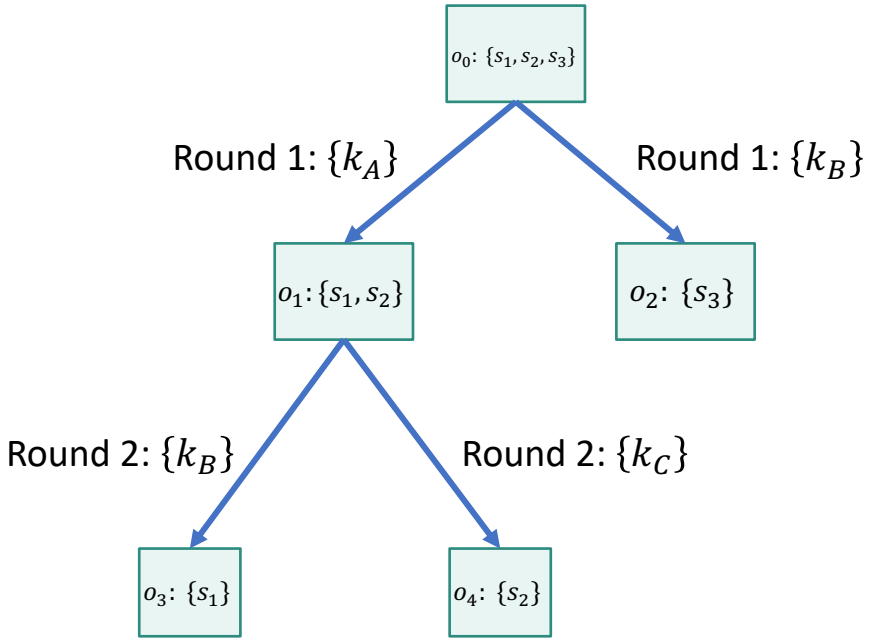
Policies Depend on Observation Histories





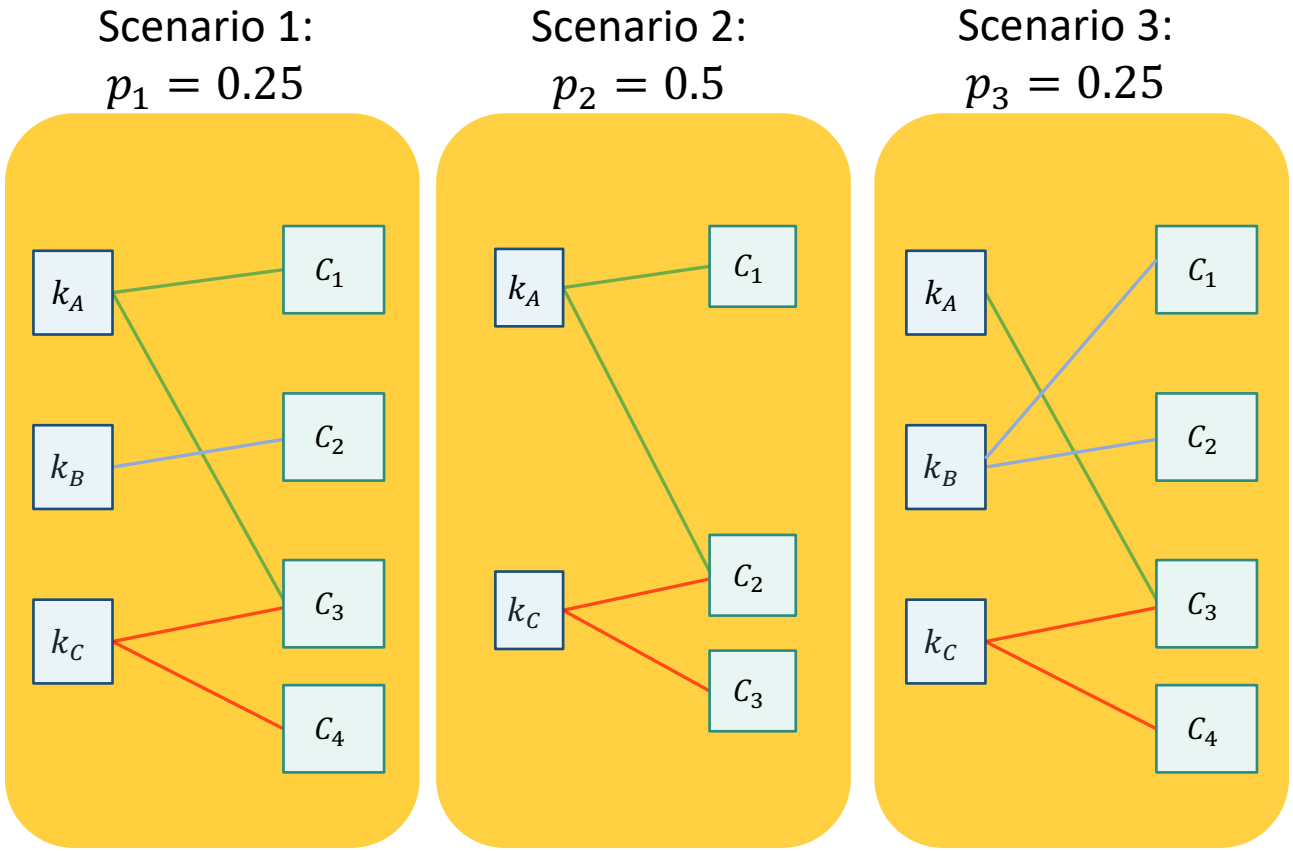
Policies Depend on Observation Histories

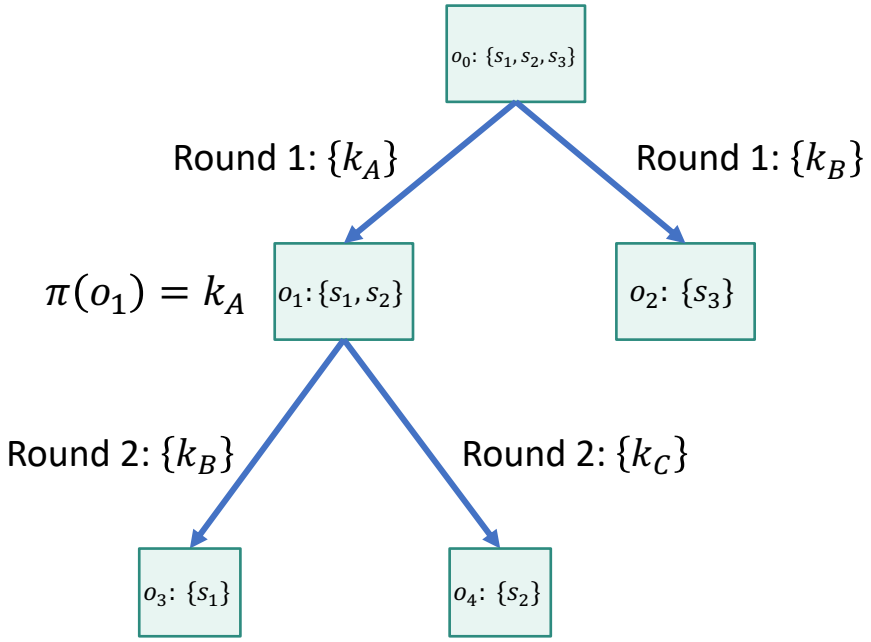




Policy π plays keys at different *infosets.*

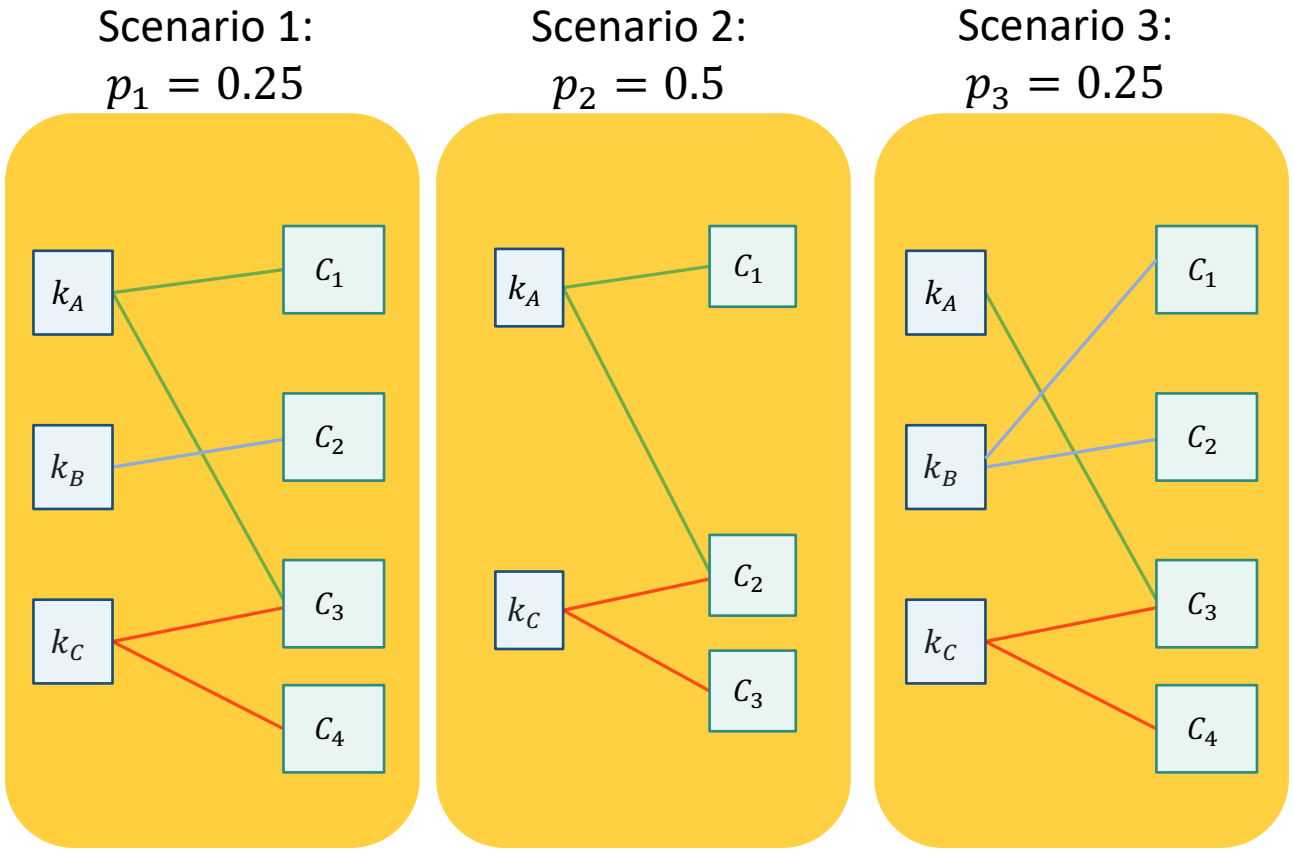
Policies Depend on Observation Histories

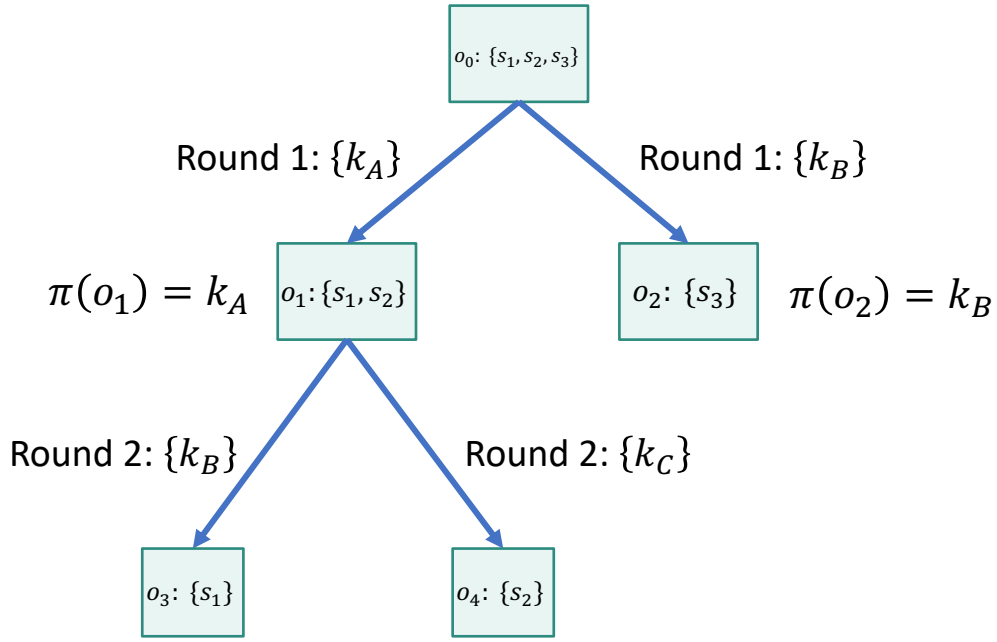




Policy π plays keys at different *infosets.*

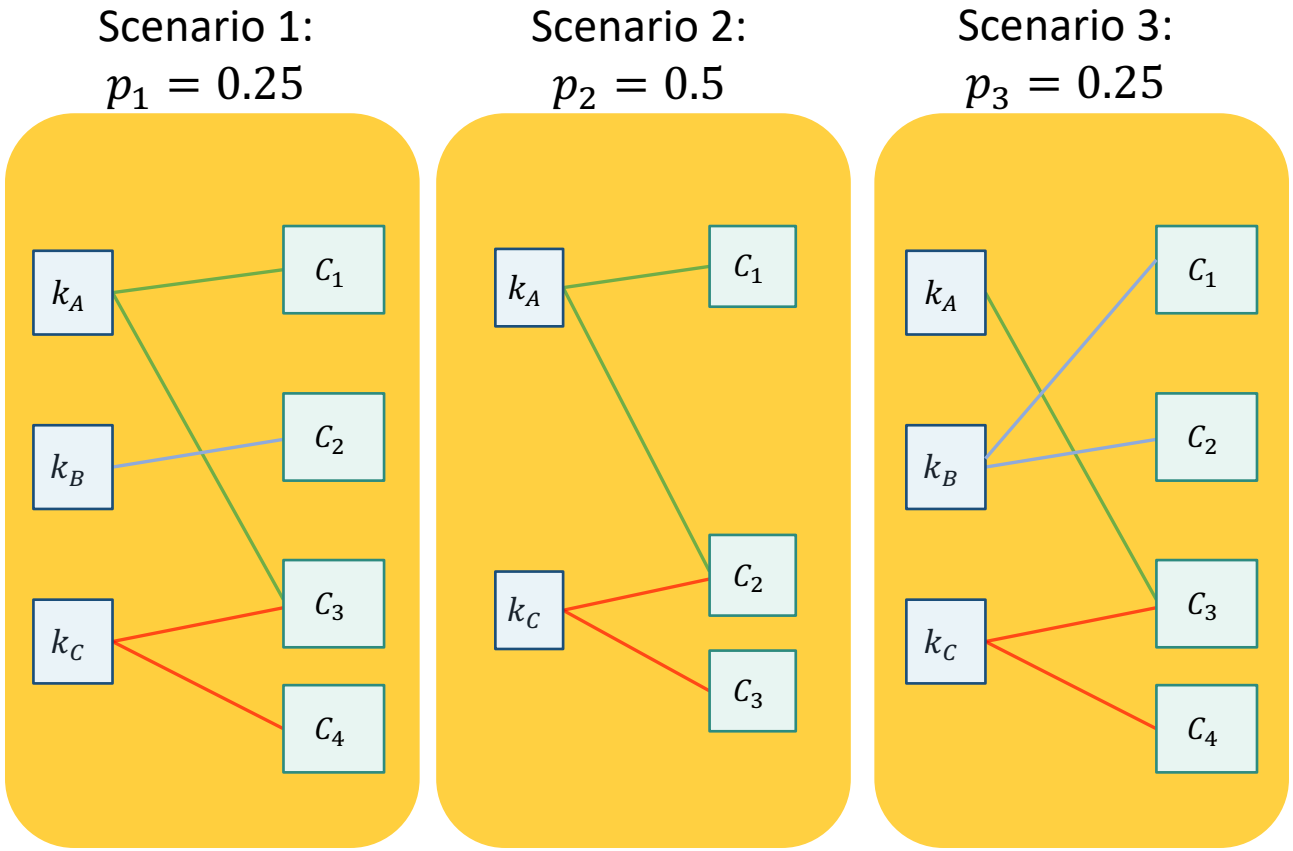
Policies Depend on Observation Histories

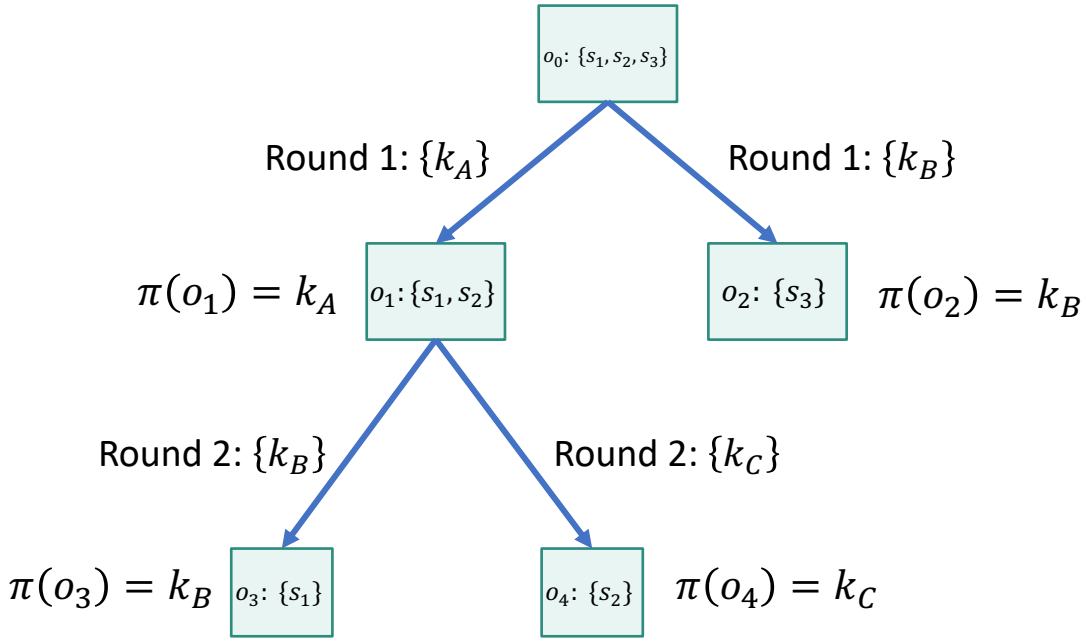




Policy π plays keys at different *infosets.*

Policies Depend on Observation Histories





Policy π plays keys at different *infosets.*

Fixed-Order Probabilistic Scenarios

Fixed-Order Probabilistic Scenarios

Theorem:
Approximation-preserving poly-time reduction from Fixed-Order Probabilistic Scenarios to Maximum Weight Laminar Matching.

Fixed-Order Probabilistic Scenarios

Theorem:
Approximation-preserving poly-time reduction from Fixed-Order Probabilistic Scenarios to Maximum Weight Laminar Matching.

Corollary:
Poly-time $\left(1 - \frac{1}{e}\right)$-approximation to Bayes-optimal fixed-order probabilistic scenarios policy.

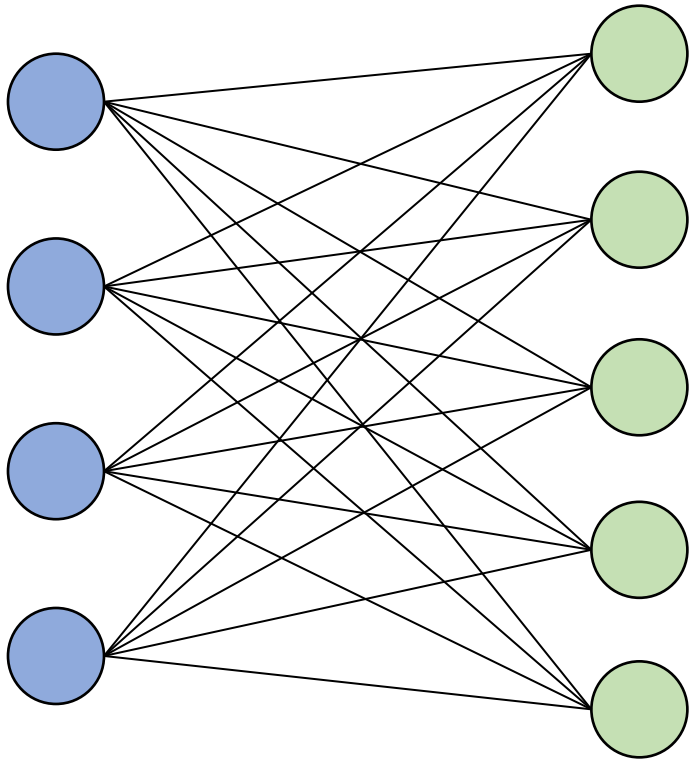
Fixed-Order Probabilistic Scenarios

Theorem:
Approximation-preserving poly-time reduction from Fixed-Order Probabilistic Scenarios to Maximum Weight Laminar Matching.

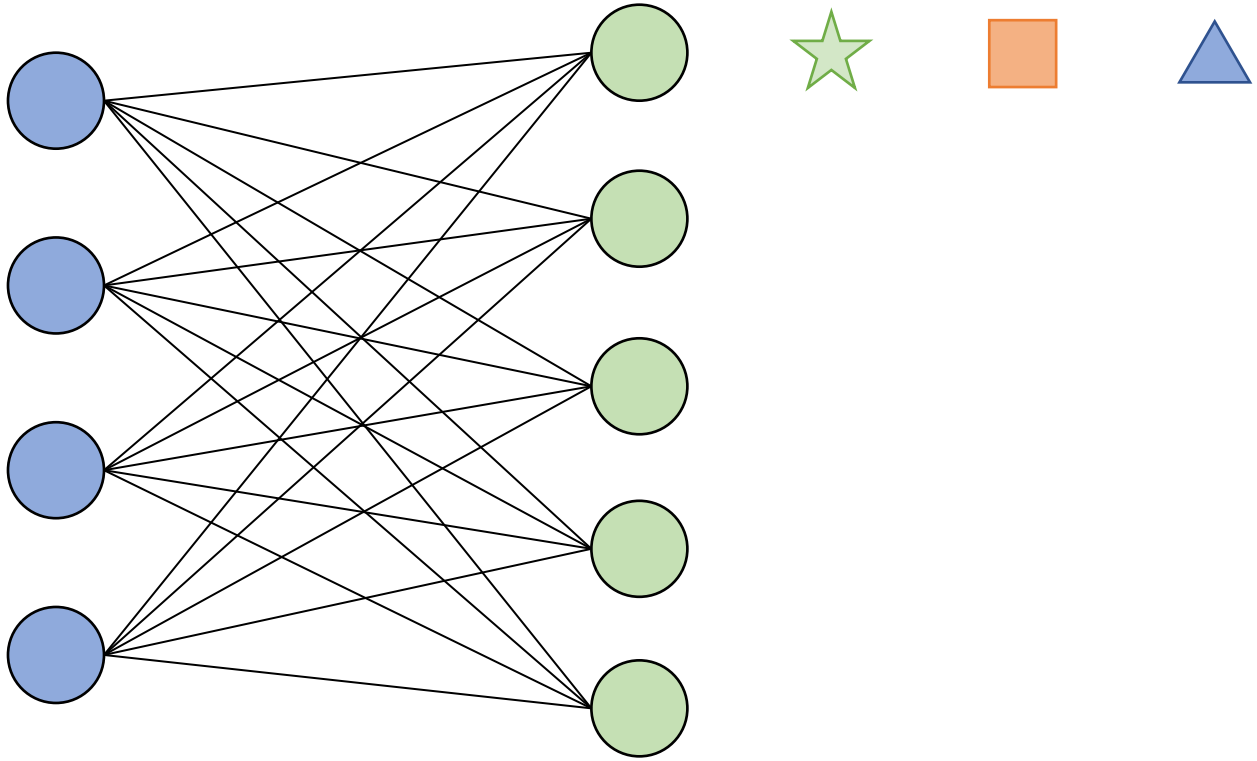
Corollary:
Poly-time $\left(1-\frac{1}{e}\right)$-approximation to Bayes-optimal fixed-order probabilistic scenarios policy.

- NP-hard to approximate a policy with value better than $0.9996 * OPT$.

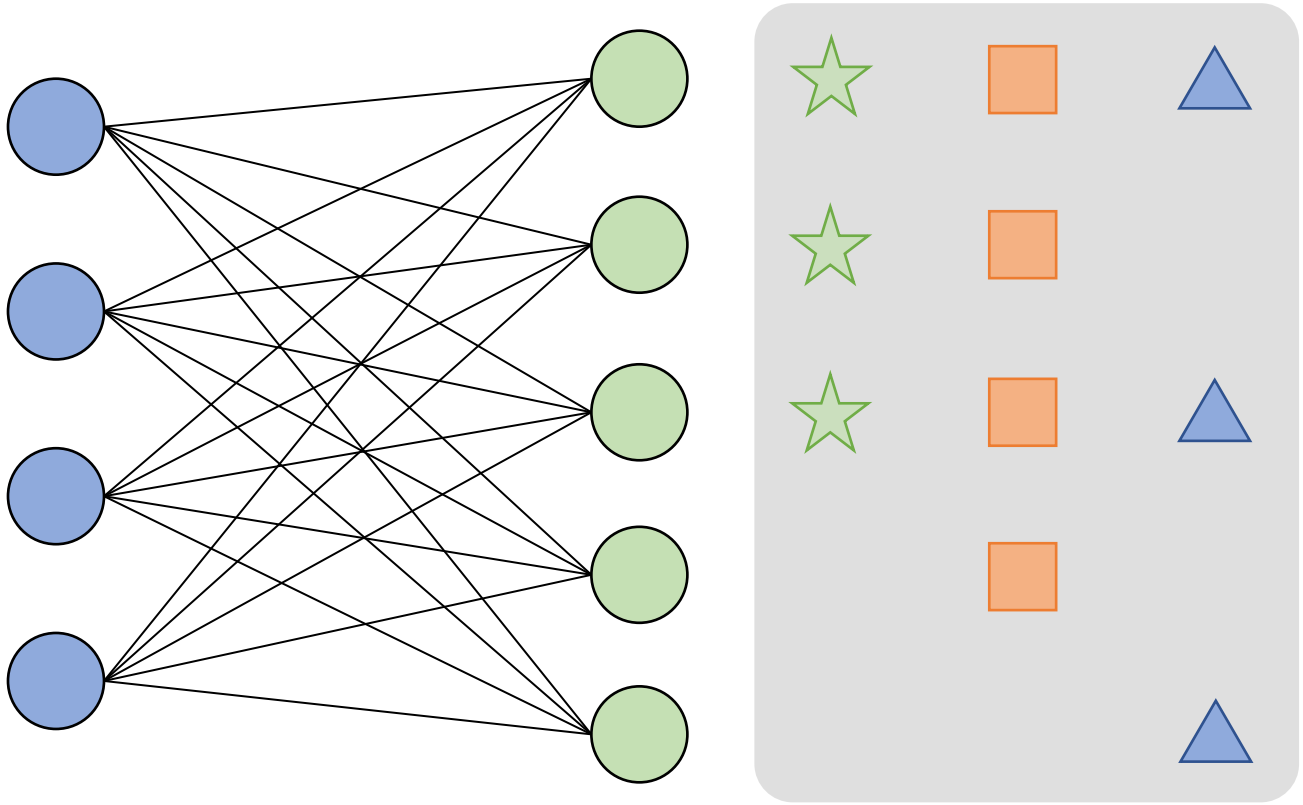
Maximum Weight Laminar Matching



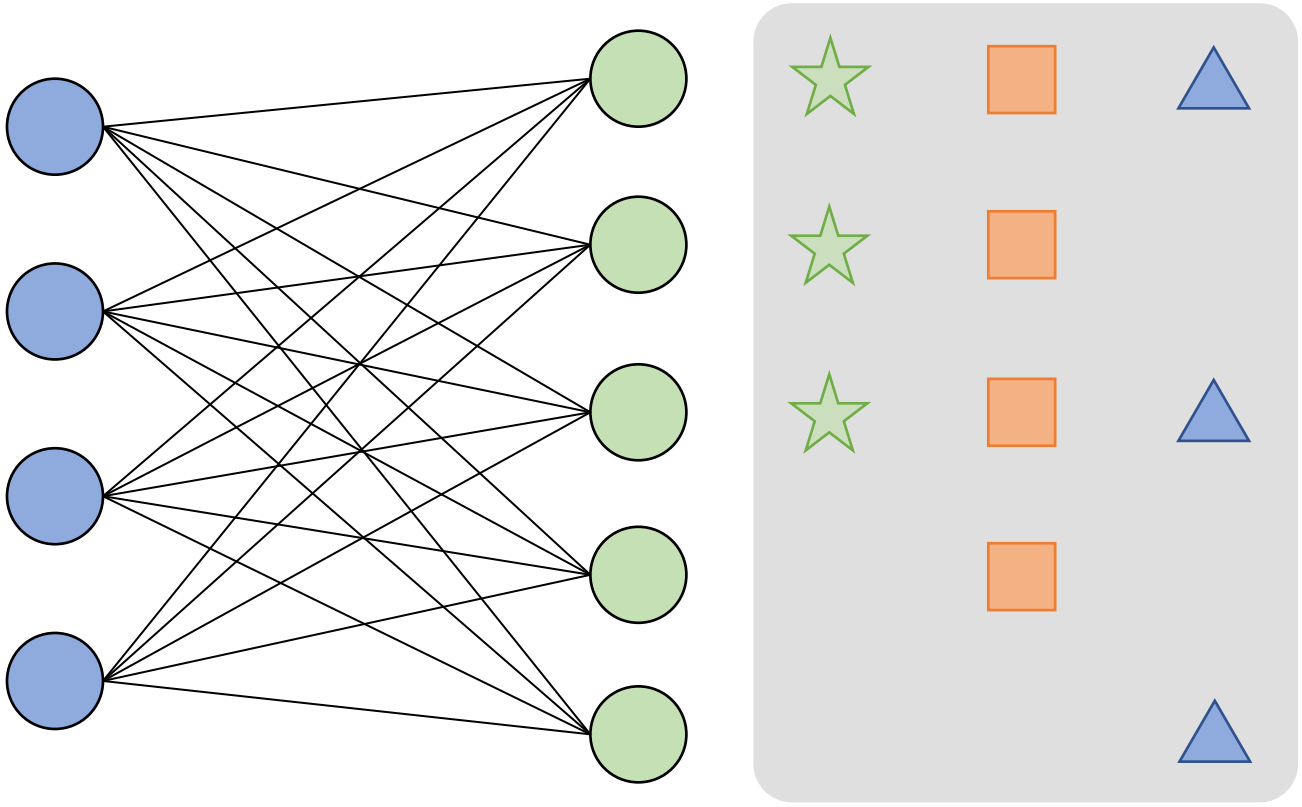
Maximum Weight Laminar Matching



Maximum Weight Laminar Matching

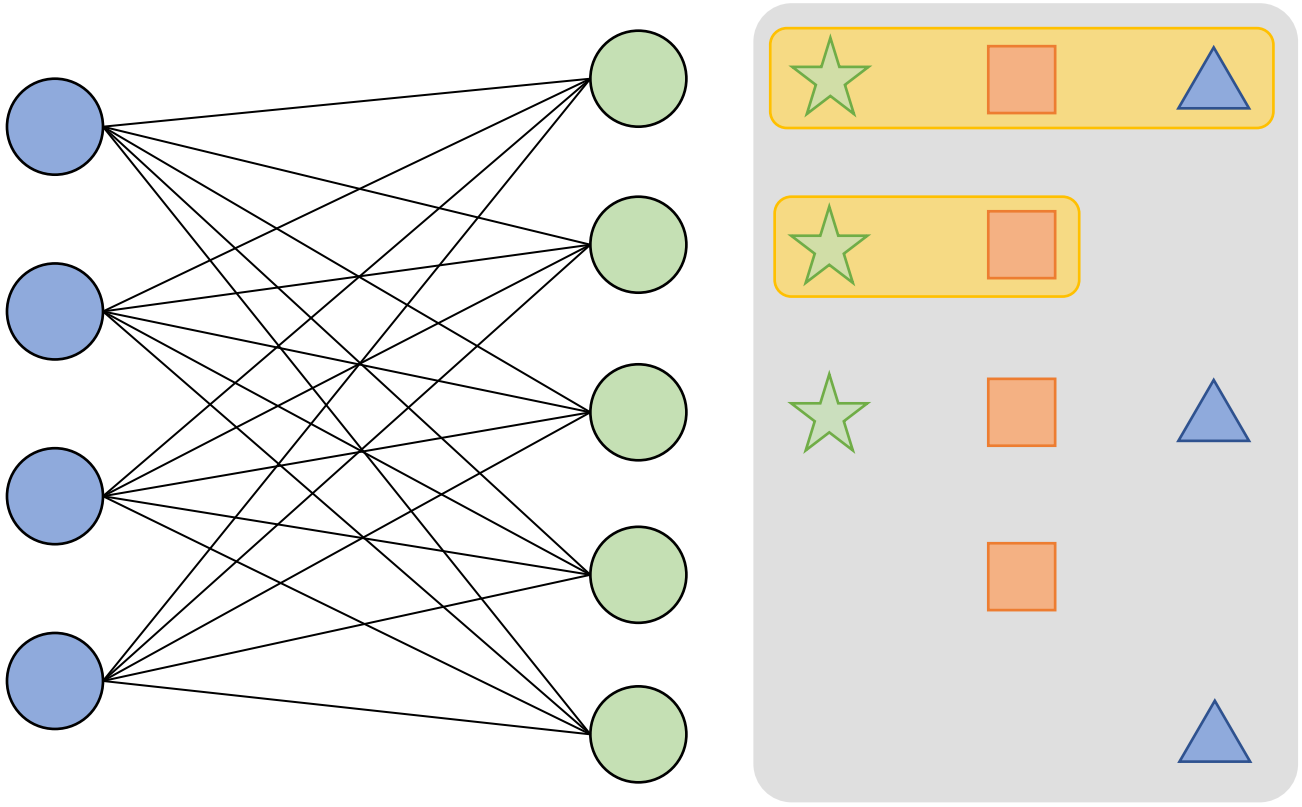


Maximum Weight Laminar Matching



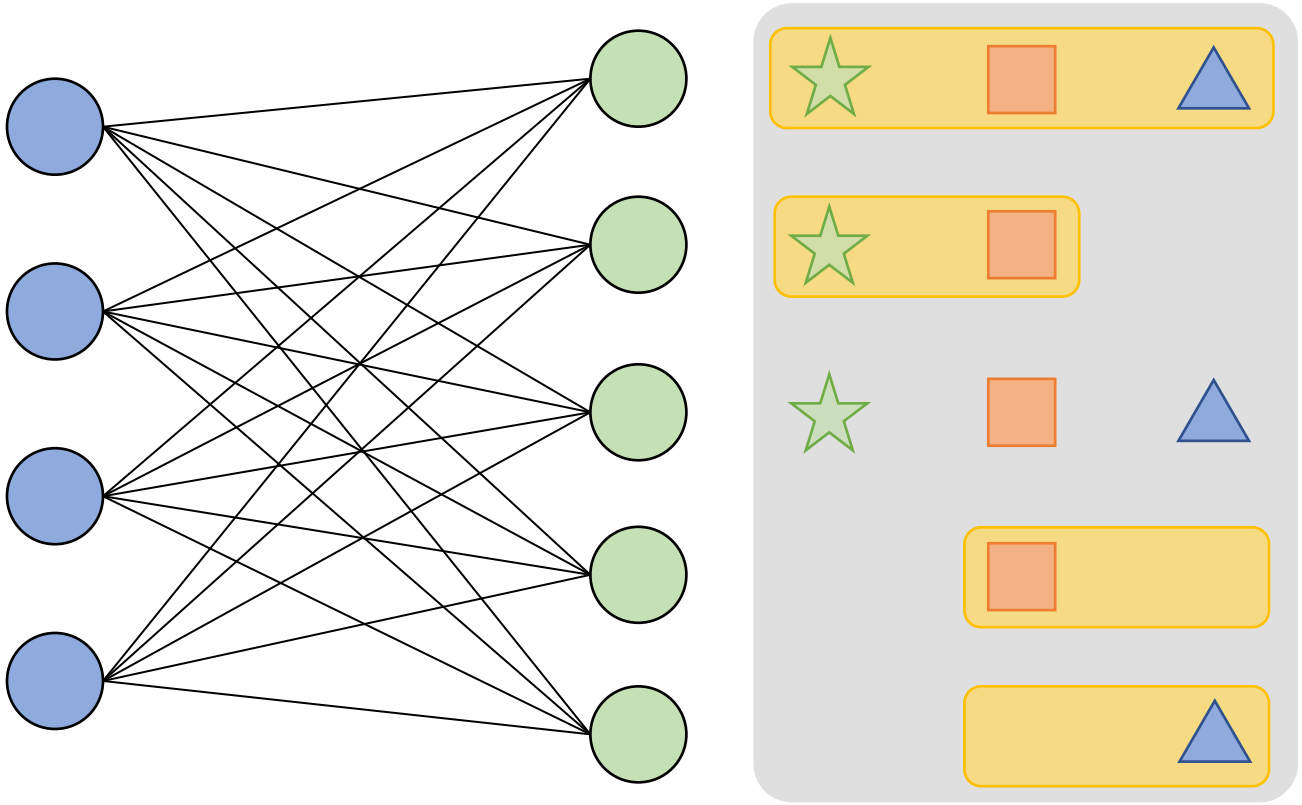
Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Maximum Weight Laminar Matching



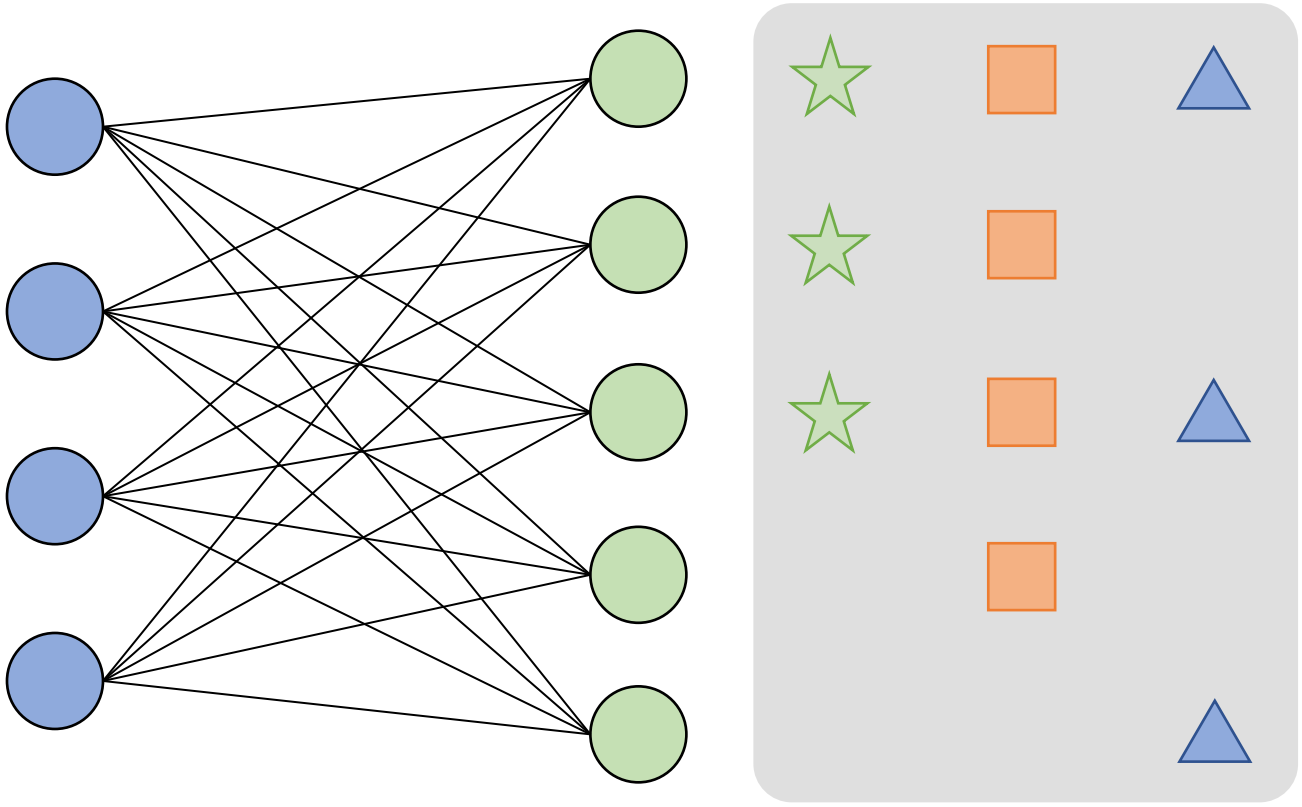
Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Maximum Weight Laminar Matching



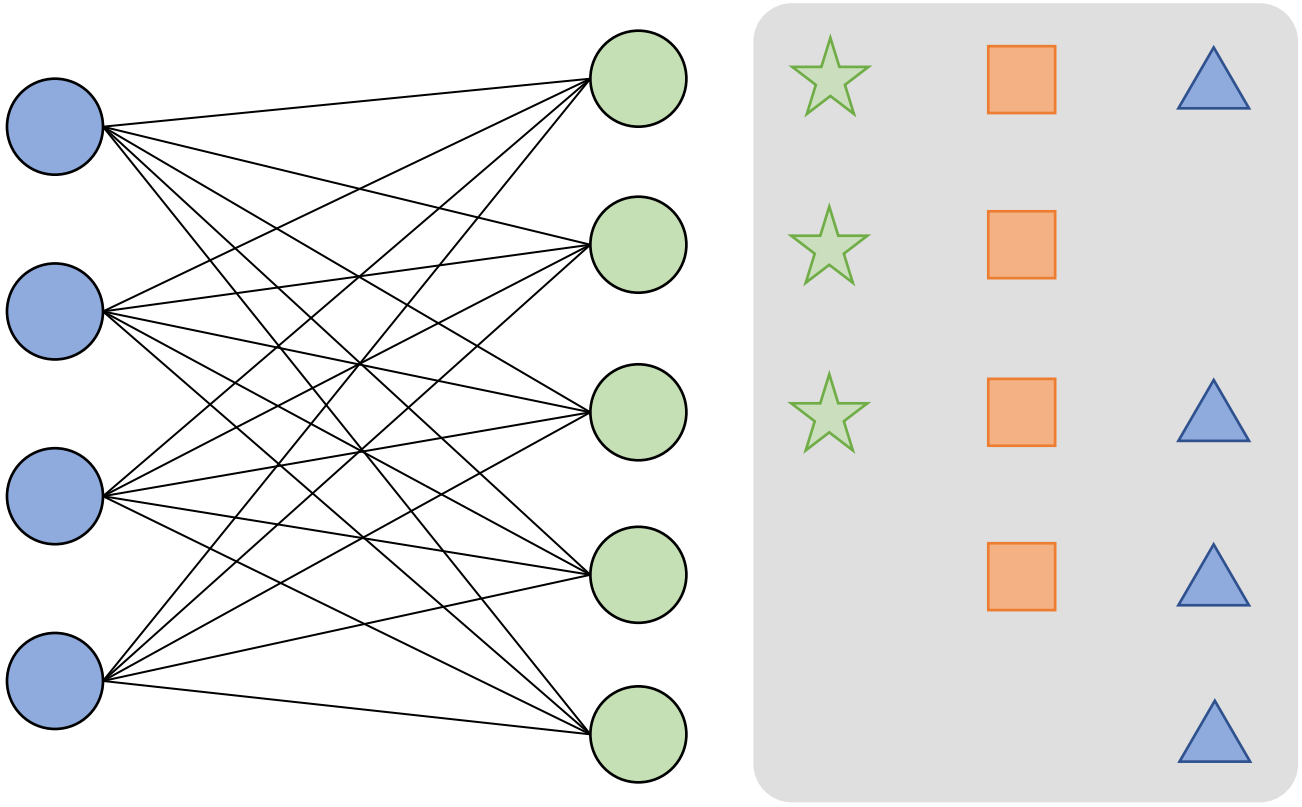
Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Maximum Weight Laminar Matching



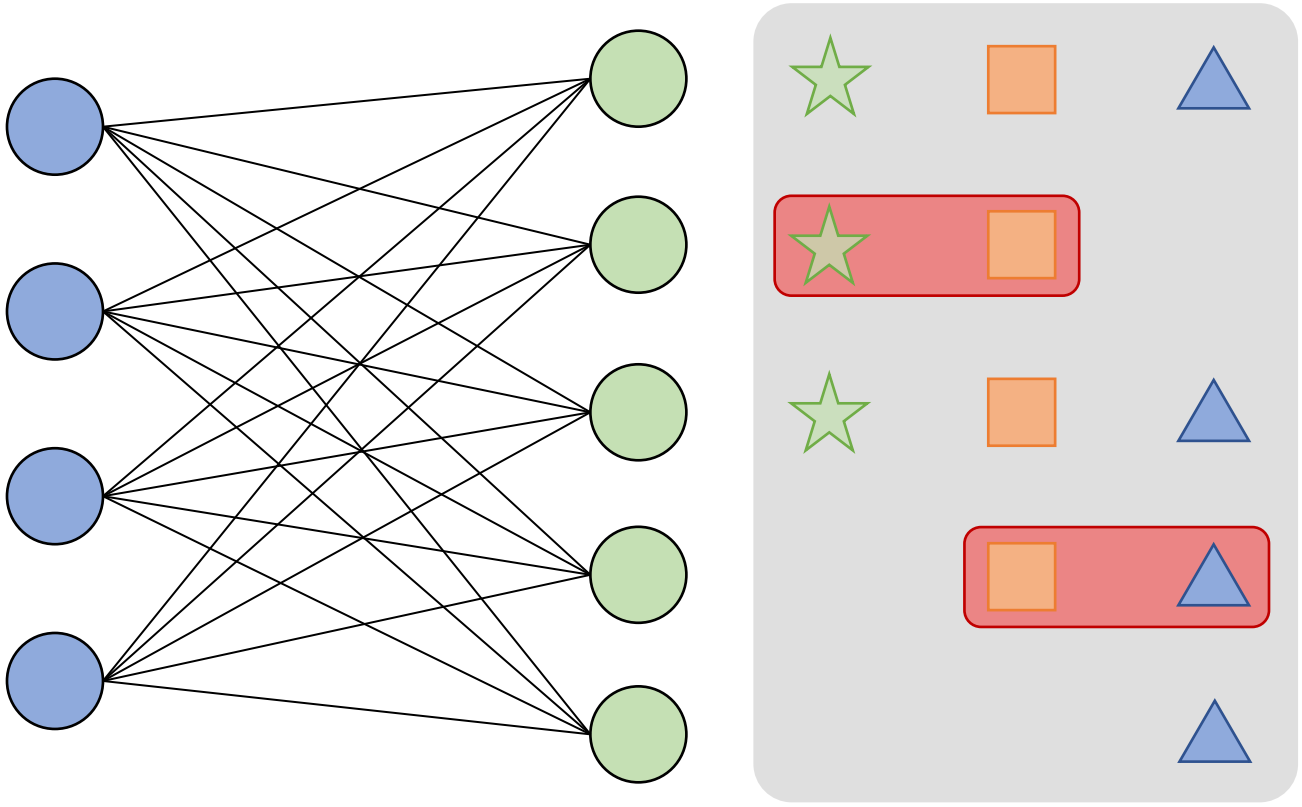
Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Maximum Weight Laminar Matching



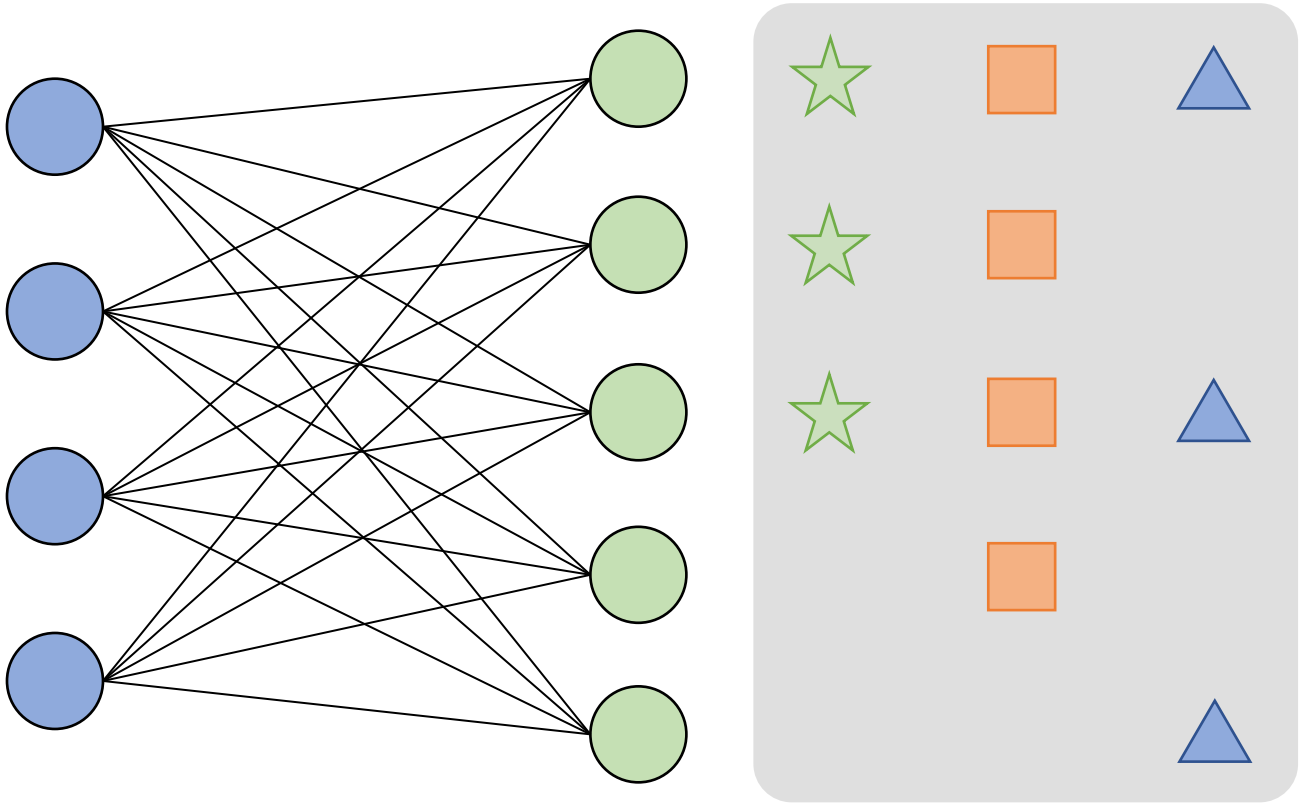
Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Maximum Weight Laminar Matching



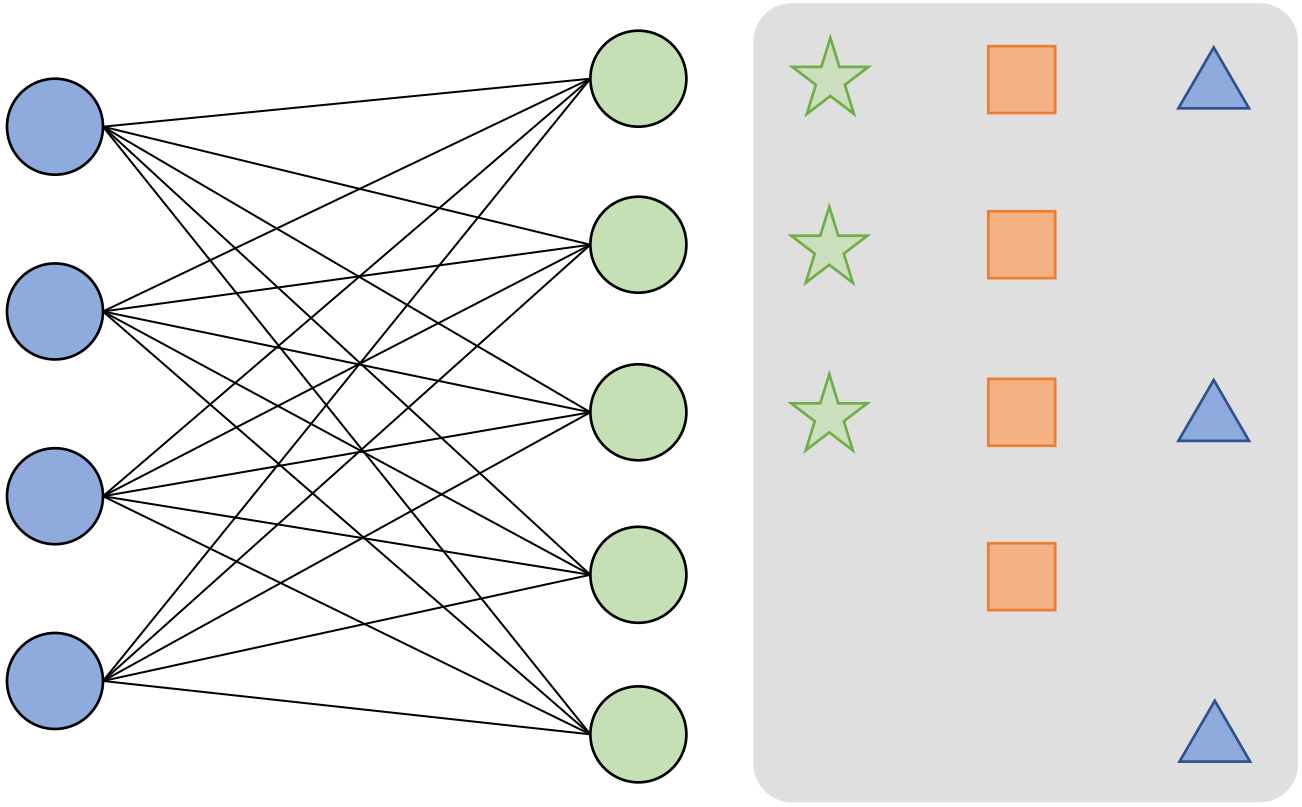
Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Maximum Weight Laminar Matching



Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

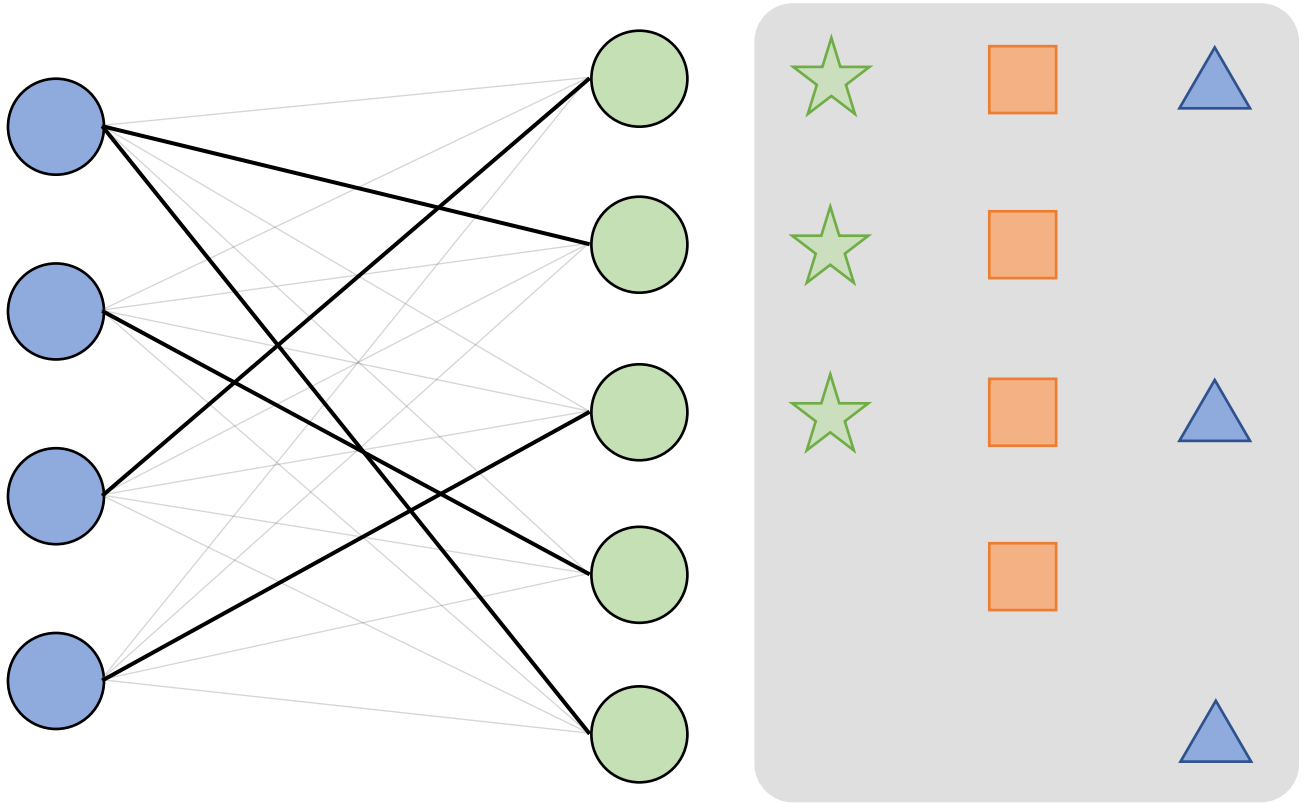
Maximum Weight Laminar Matching



Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Laminar Matching:
Selected edge set M must simultaneously be a matching on every type-restricted subgraph.

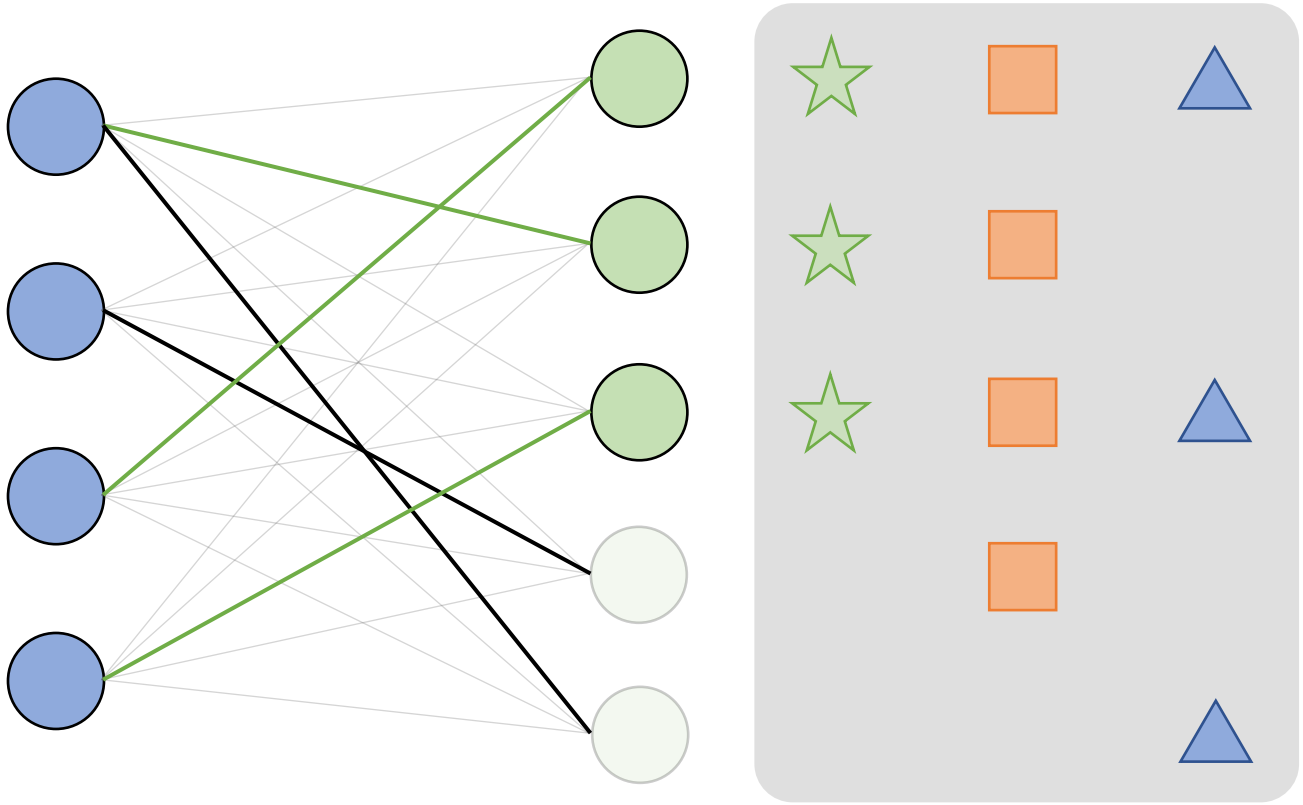
Maximum Weight Laminar Matching



Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Laminar Matching:
Selected edge set M must simultaneously be a matching on every type-restricted subgraph.

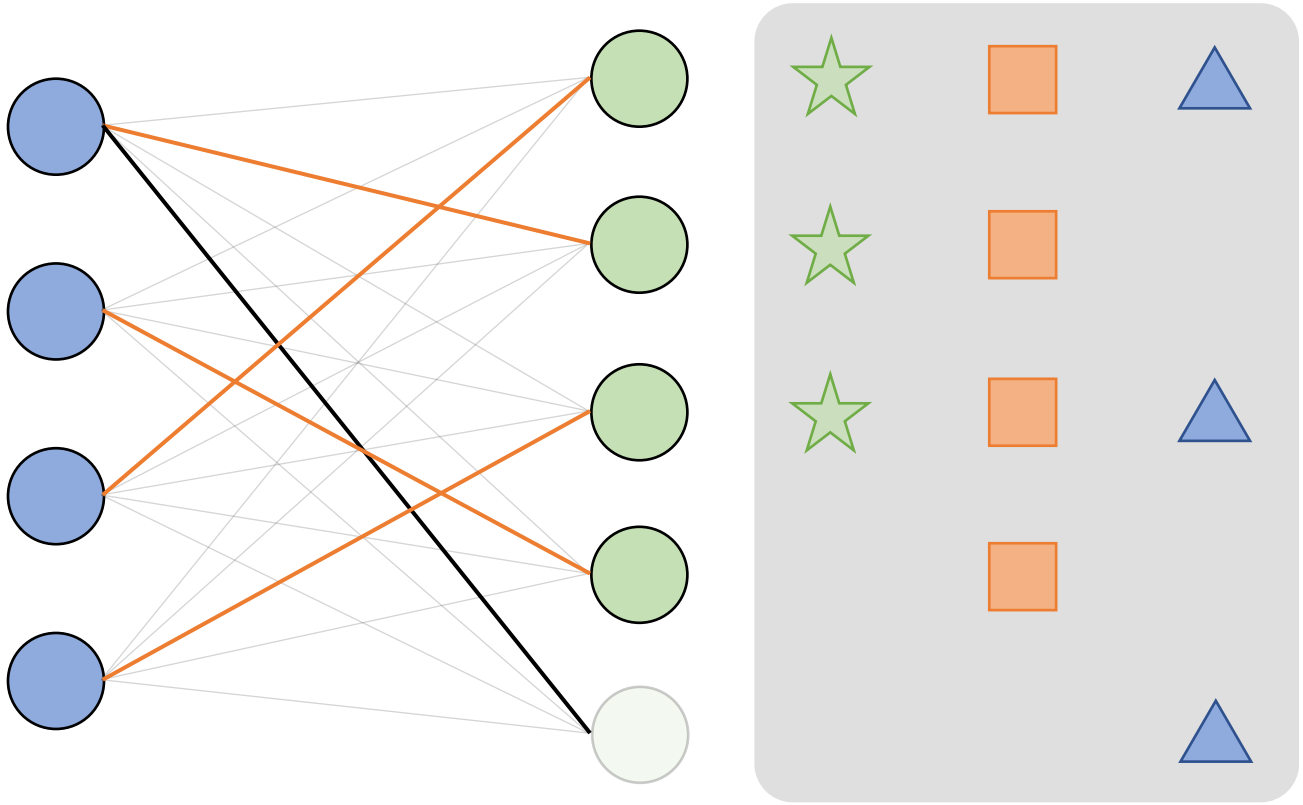
Maximum Weight Laminar Matching



Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Laminar Matching:
Selected edge set M must simultaneously be a matching on every type-restricted subgraph.

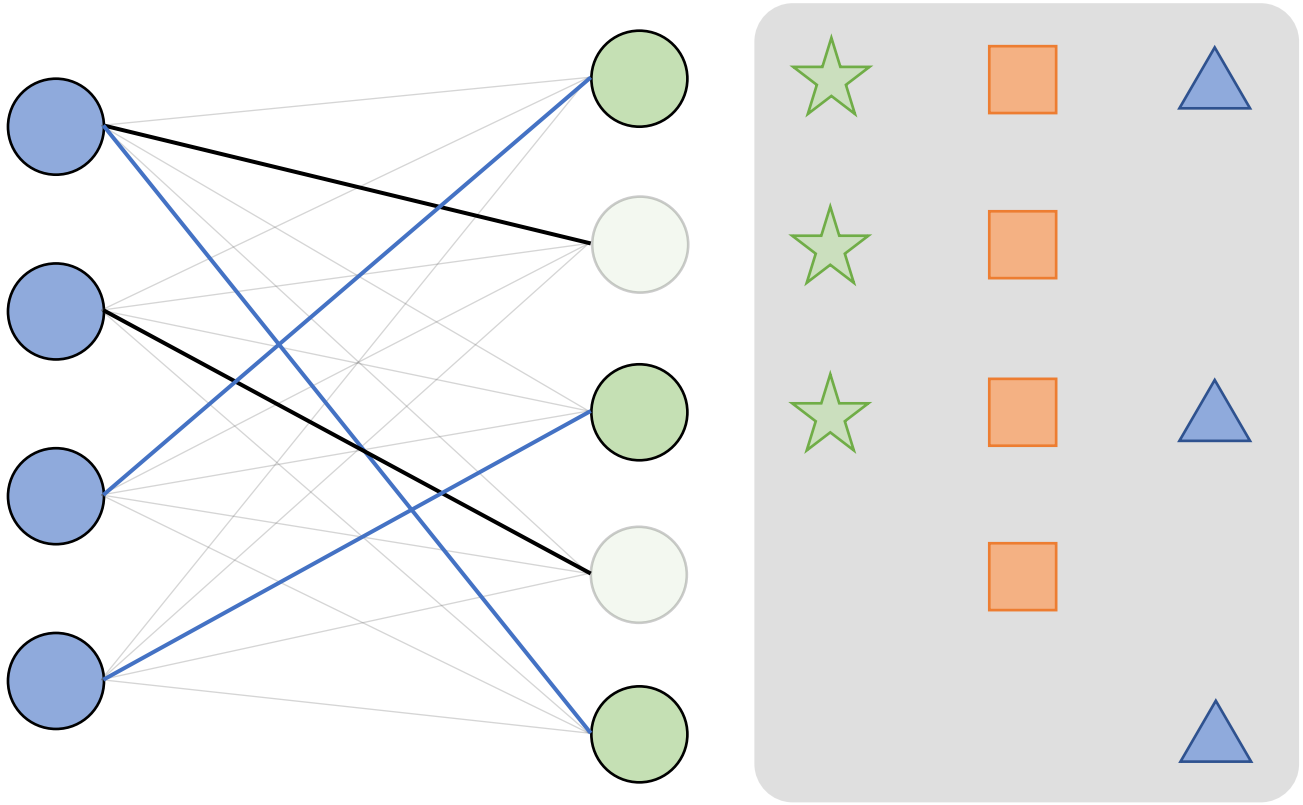
Maximum Weight Laminar Matching



Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Laminar Matching:
Selected edge set M must simultaneously be a matching on every type-restricted subgraph.

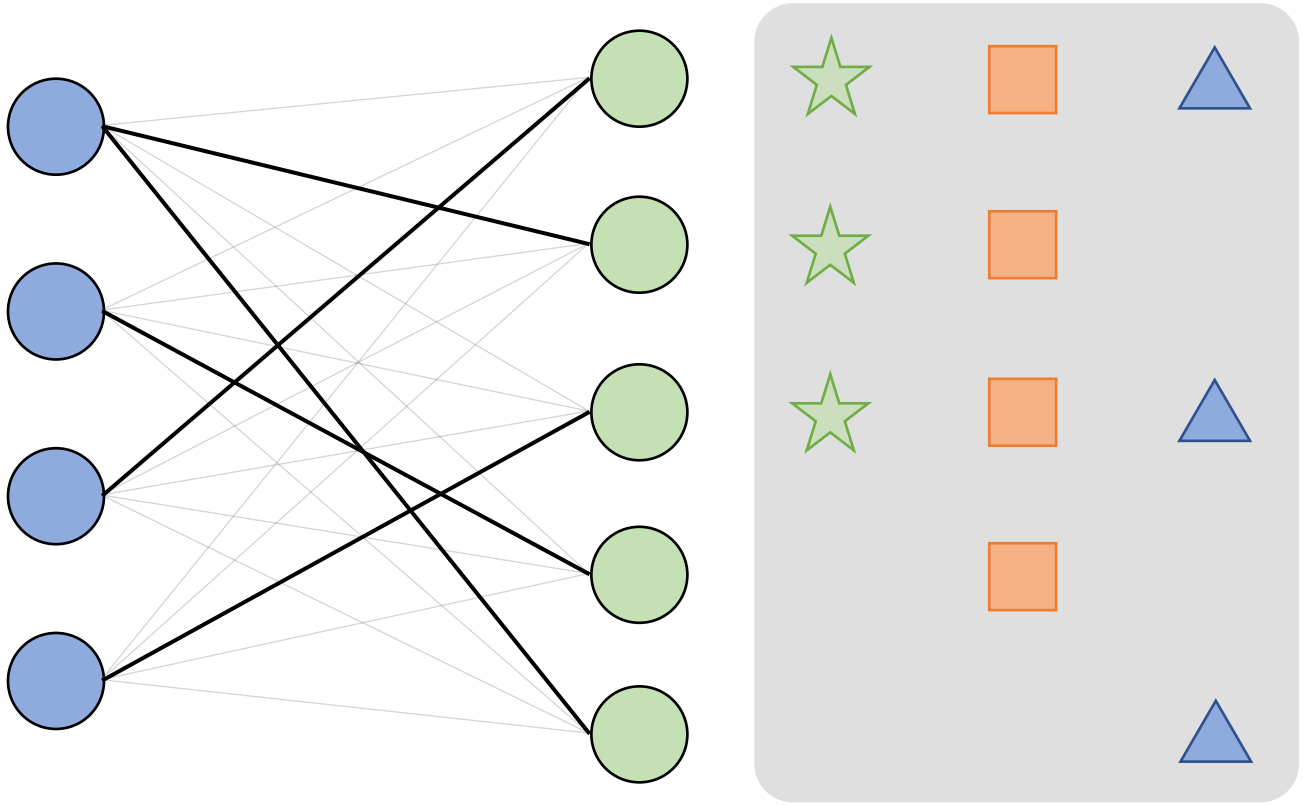
Maximum Weight Laminar Matching



Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

Laminar Matching:
Selected edge set M must simultaneously be a matching on every type-restricted subgraph.

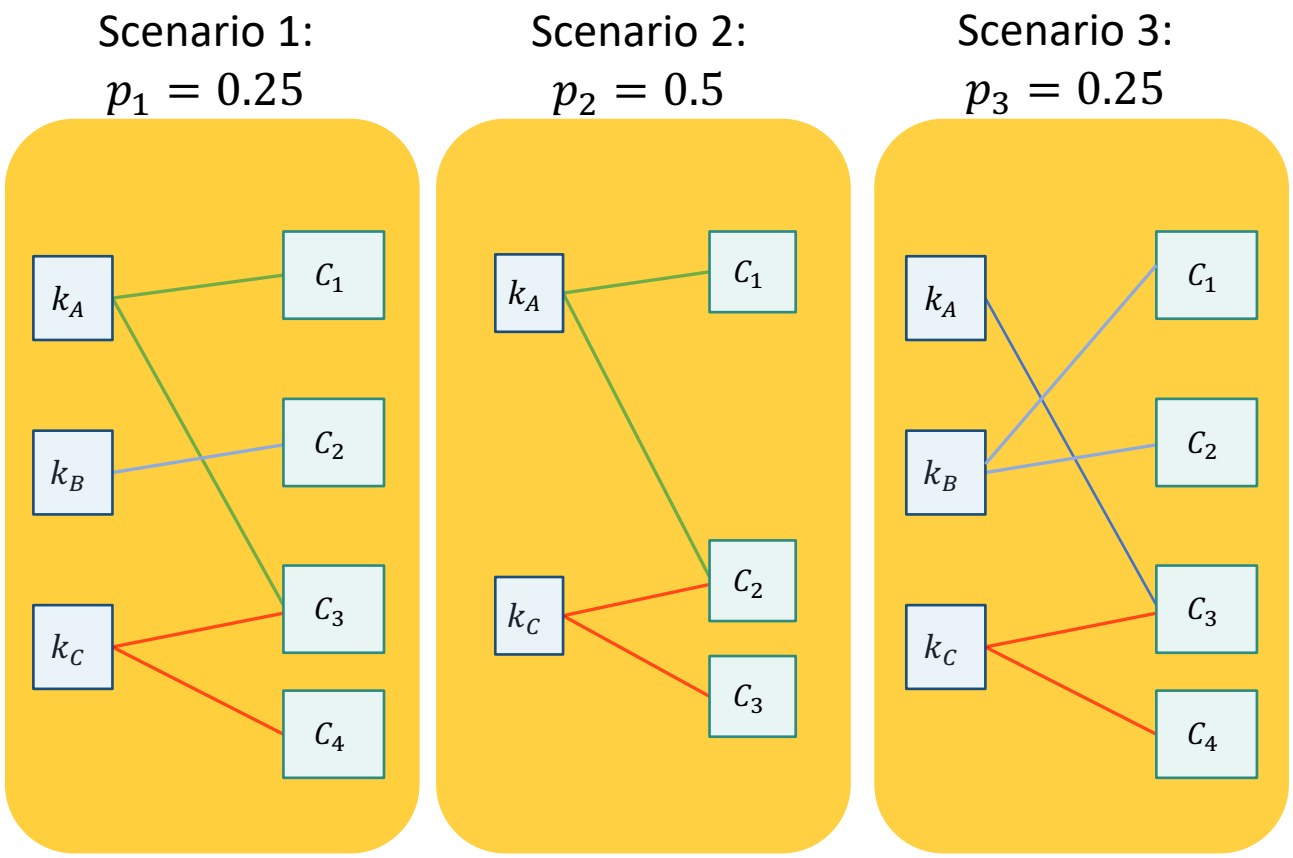
Maximum Weight Laminar Matching

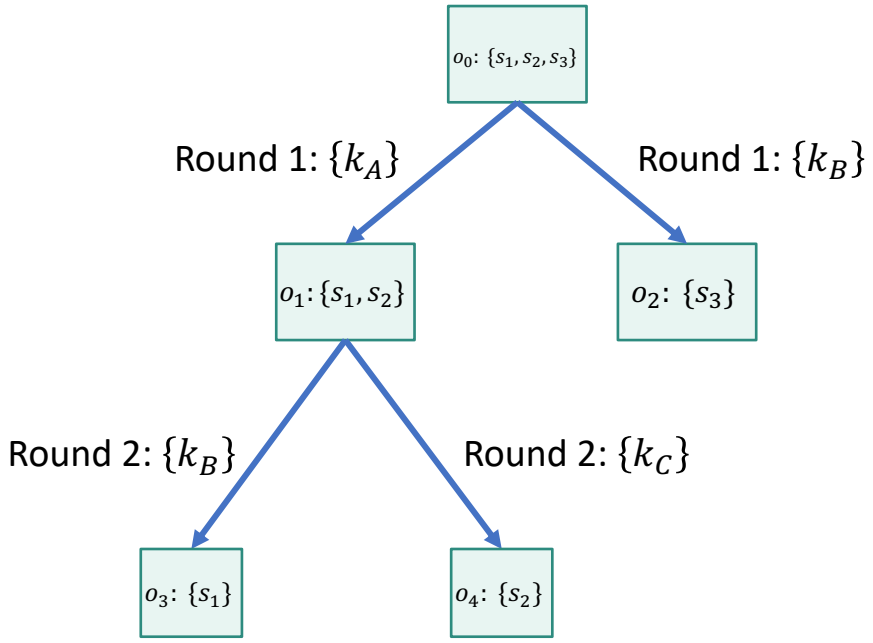


Laminar Set Family:
Type sets $T_1, \ldots, T_n \subseteq \mathcal{T}$, and for all i and j, either T_i and T_j are subsets or disjoint.

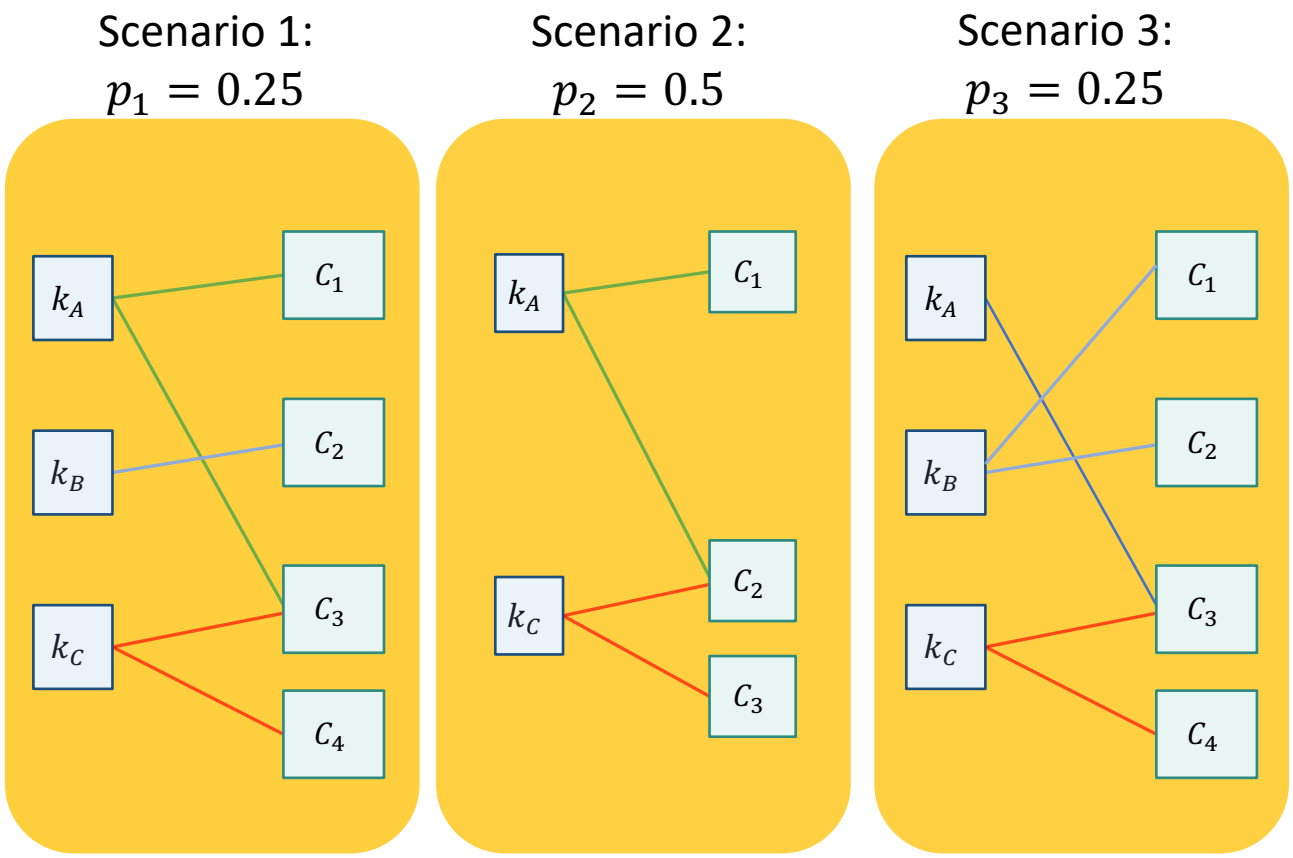
Laminar Matching:
Selected edge set M must simultaneously be a matching on every type-restricted subgraph.

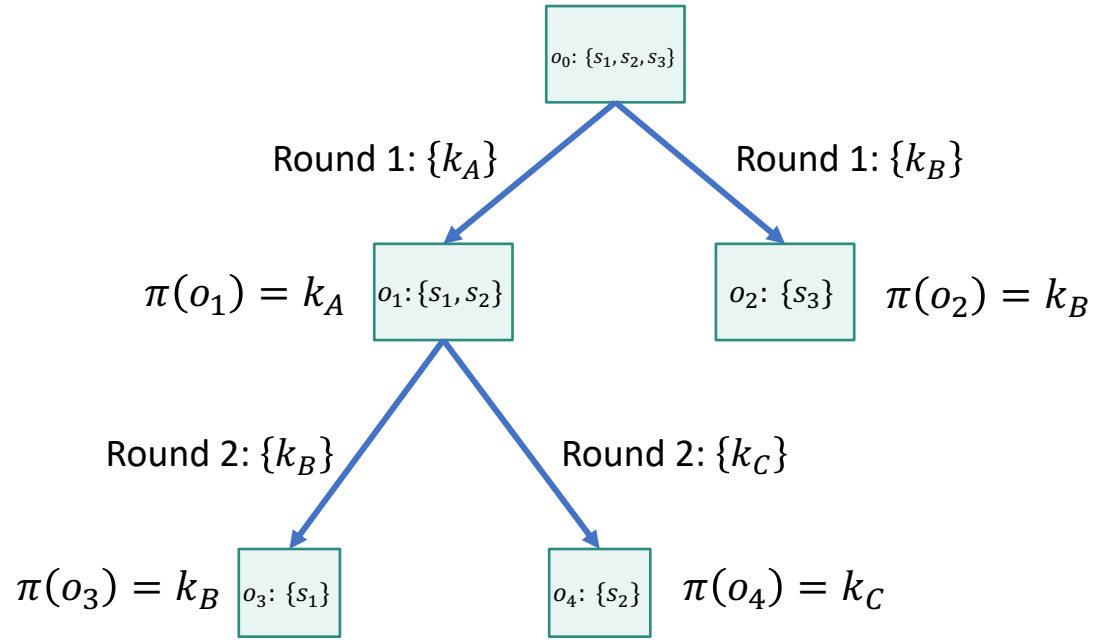
Fixed-Order Probabilistic Scenarios to MWLM



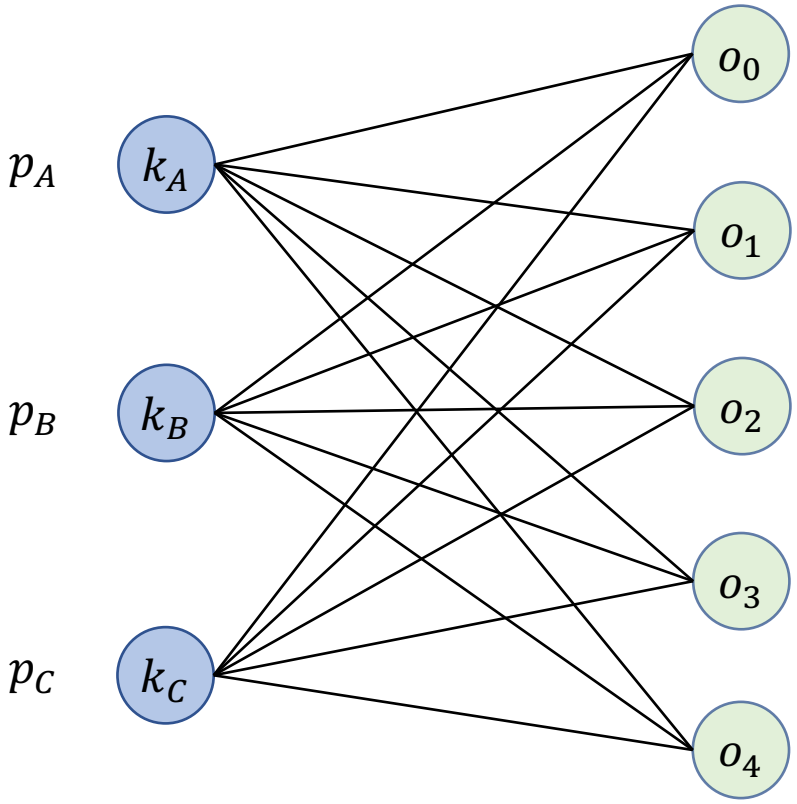


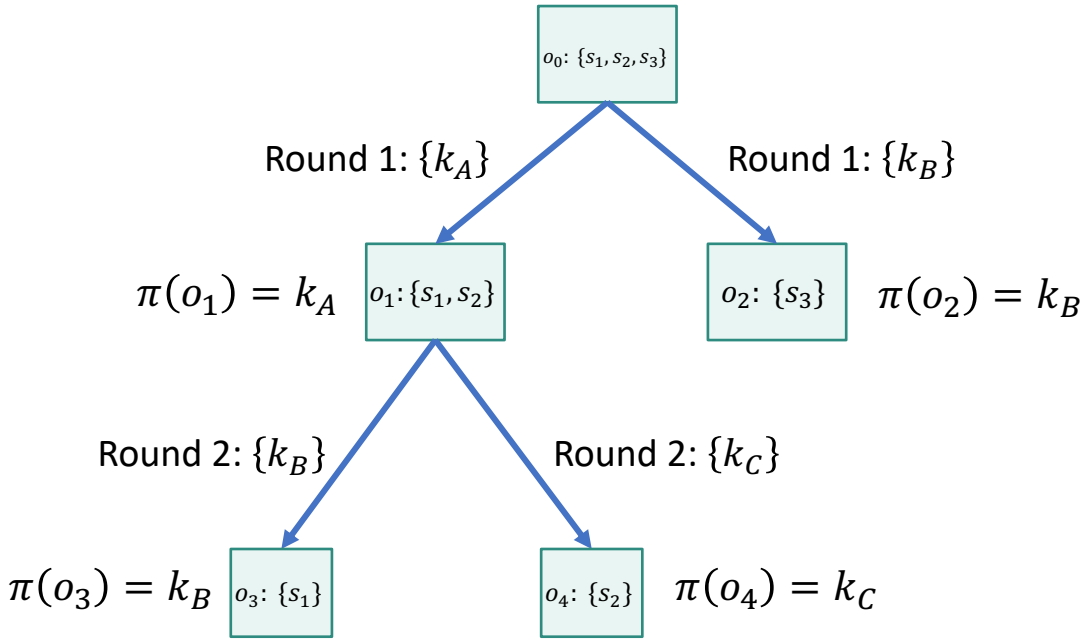
Fixed-Order Probabilistic Scenarios to MWLM



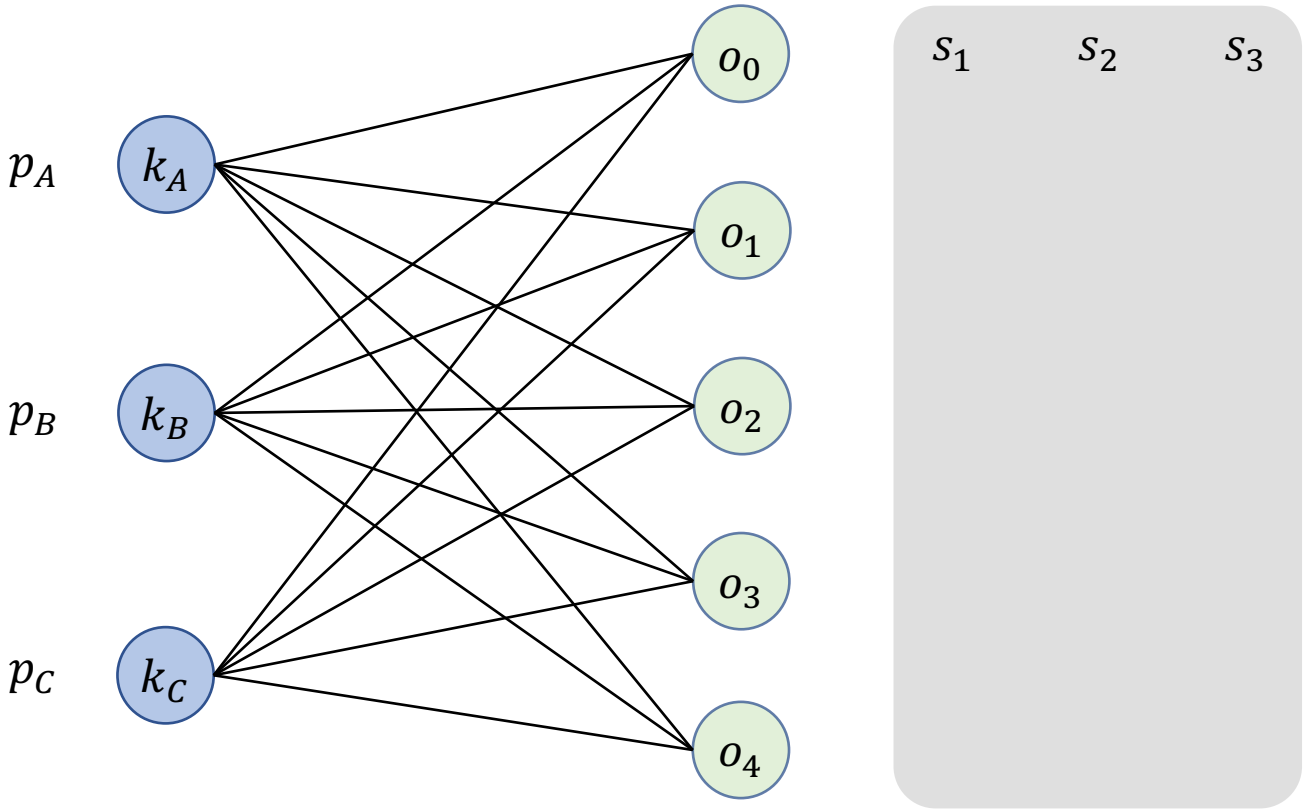


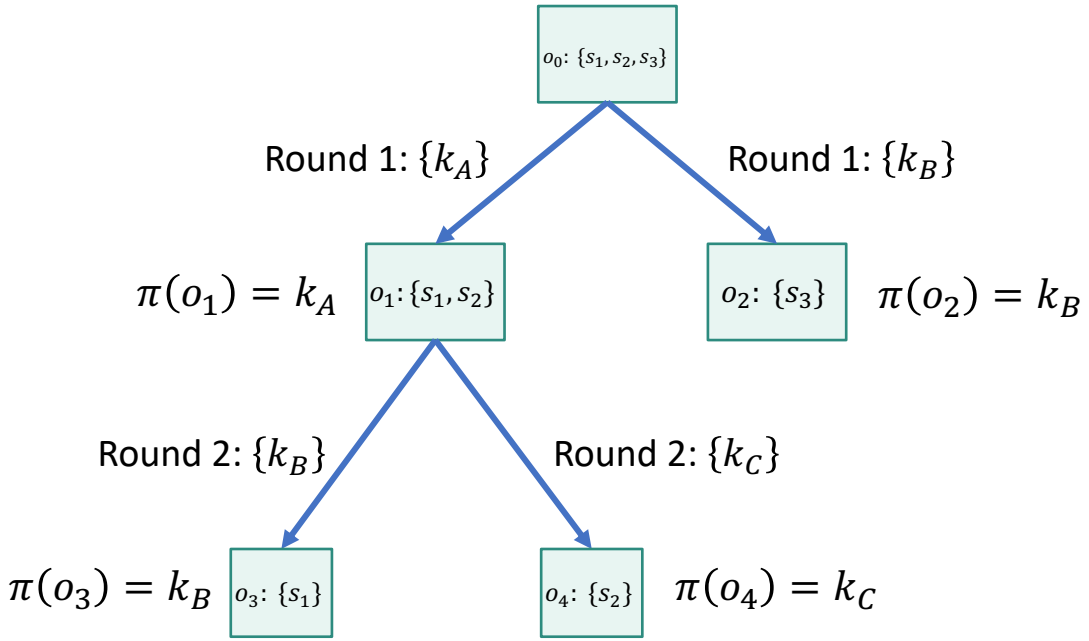
Fixed-Order Probabilistic Scenarios to MWLM



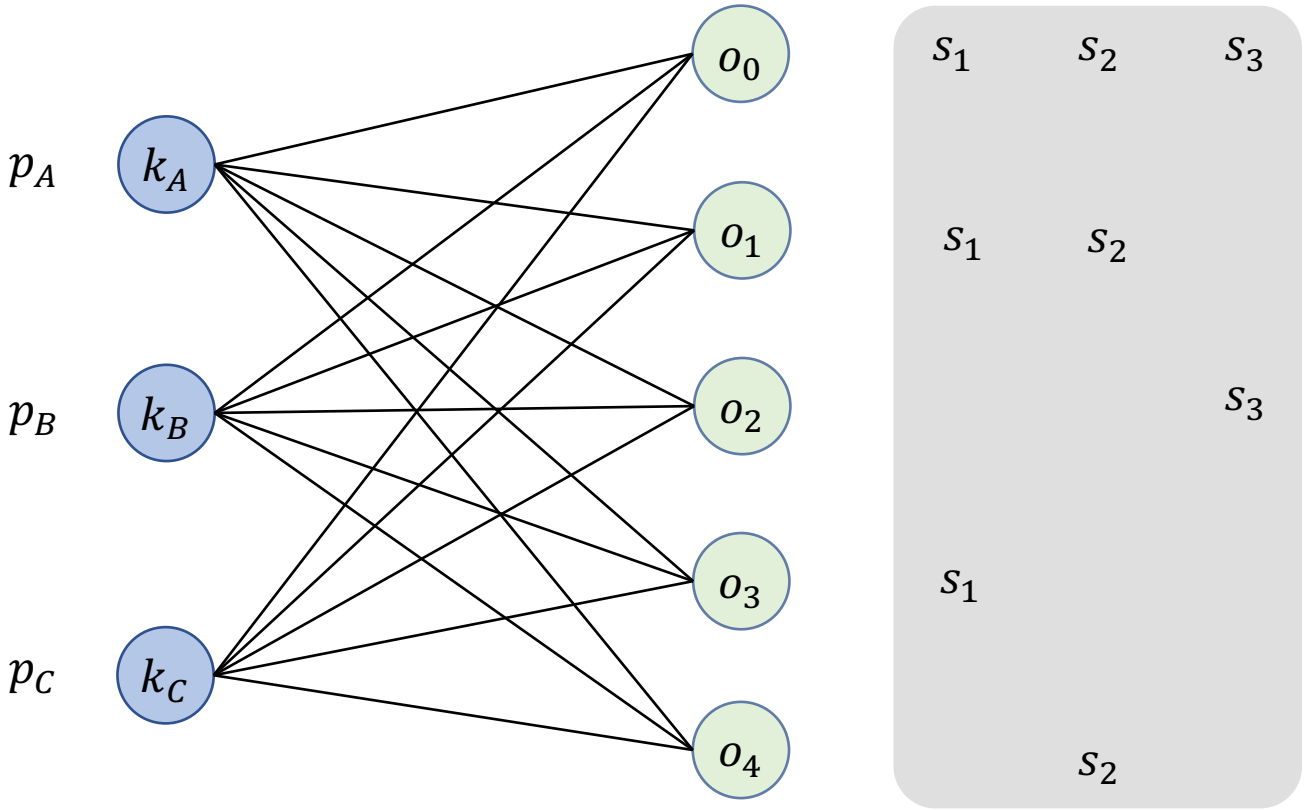


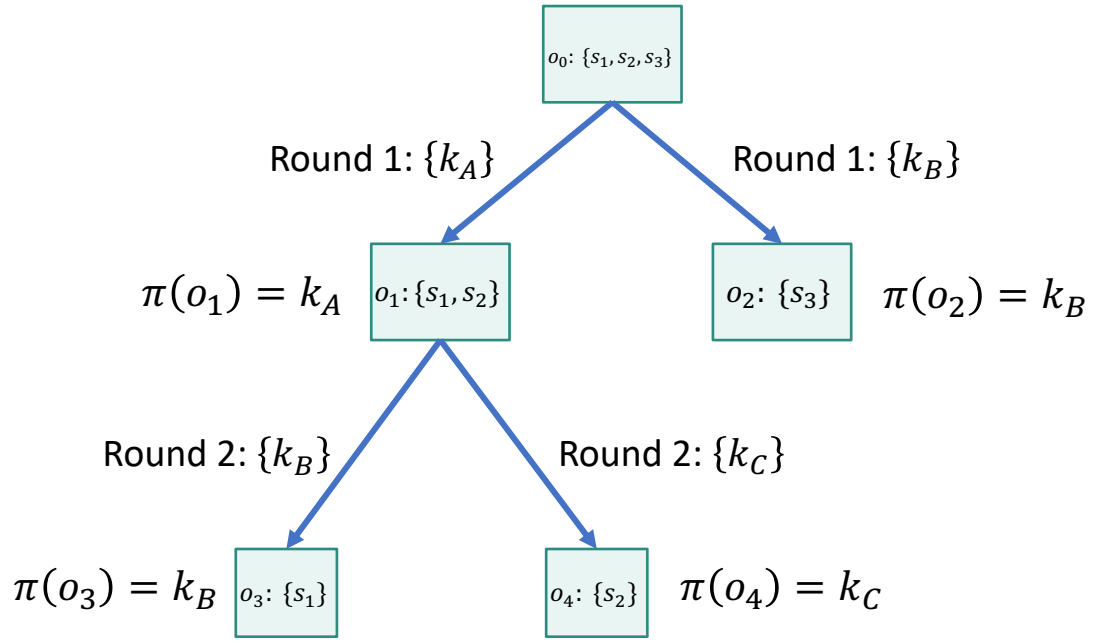
Fixed-Order Probabilistic Scenarios to MWLM



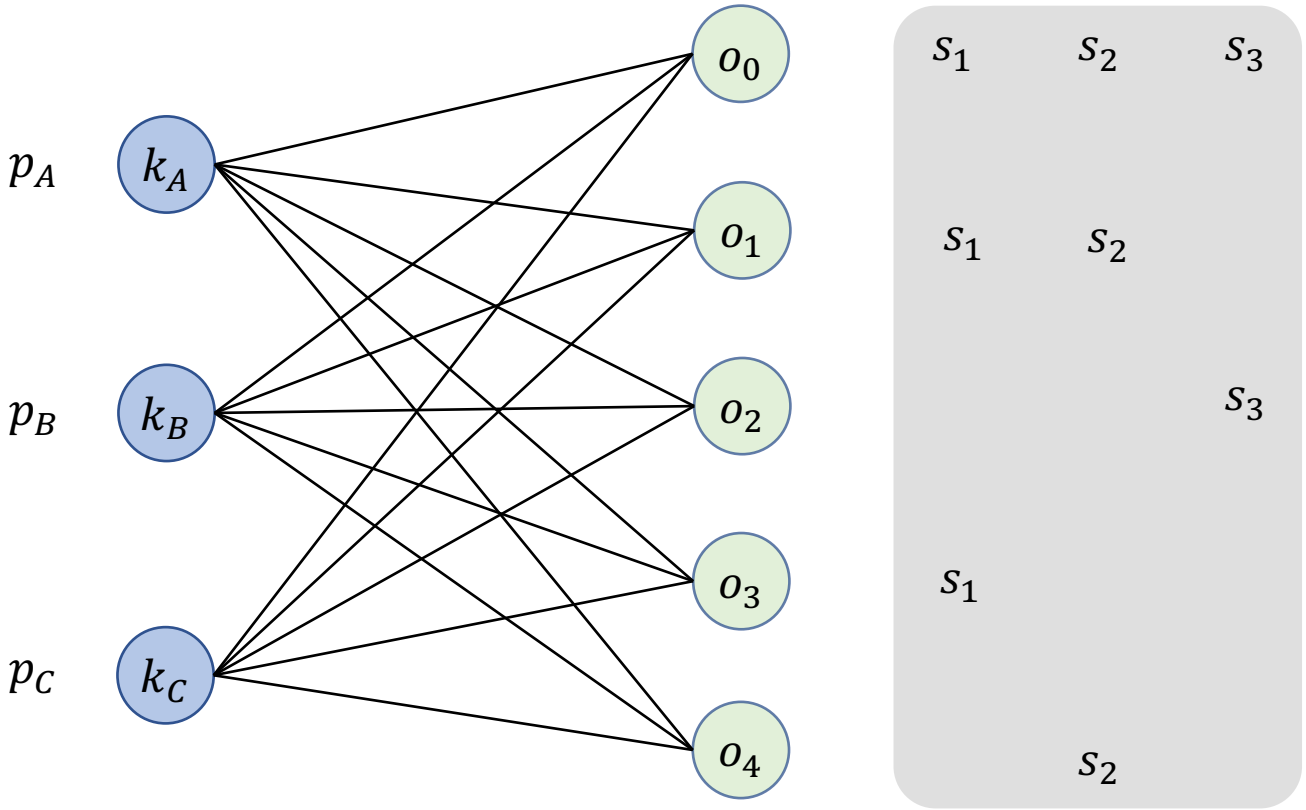


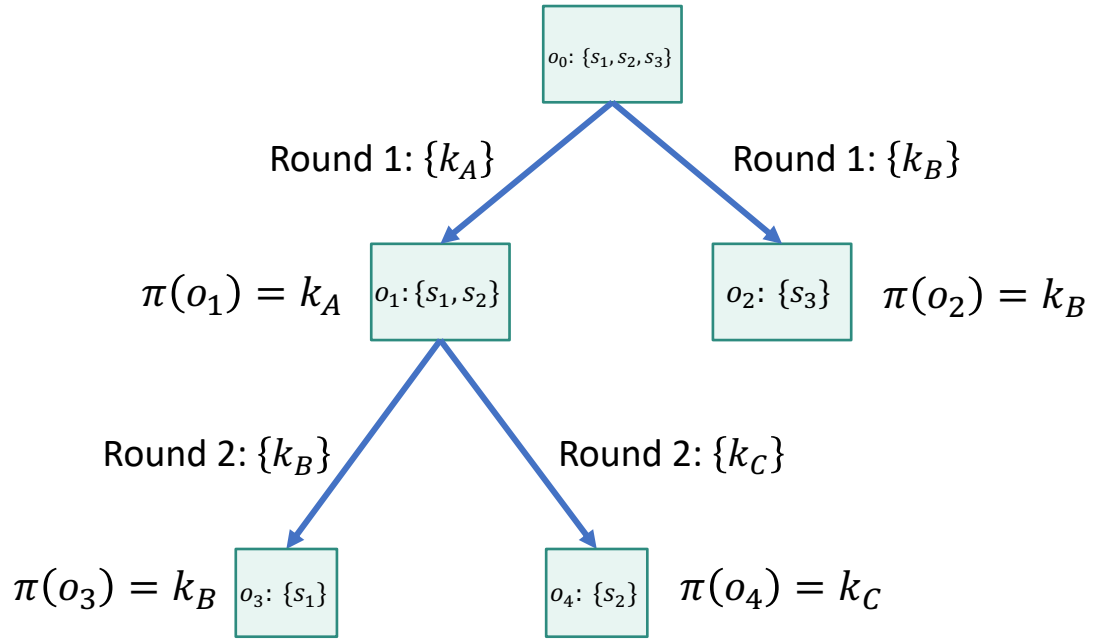
Fixed-Order Probabilistic Scenarios to MWLM





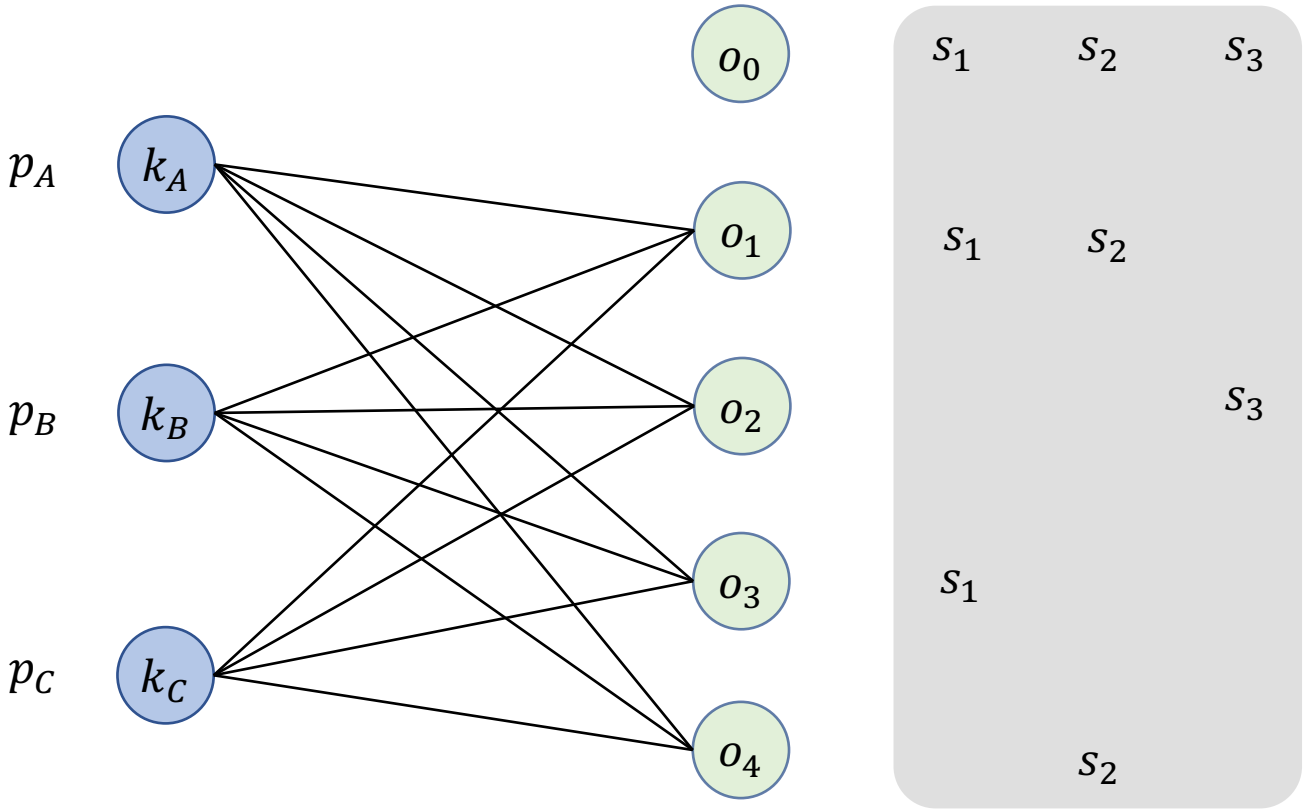
Fixed-Order Probabilistic Scenarios to MWLM



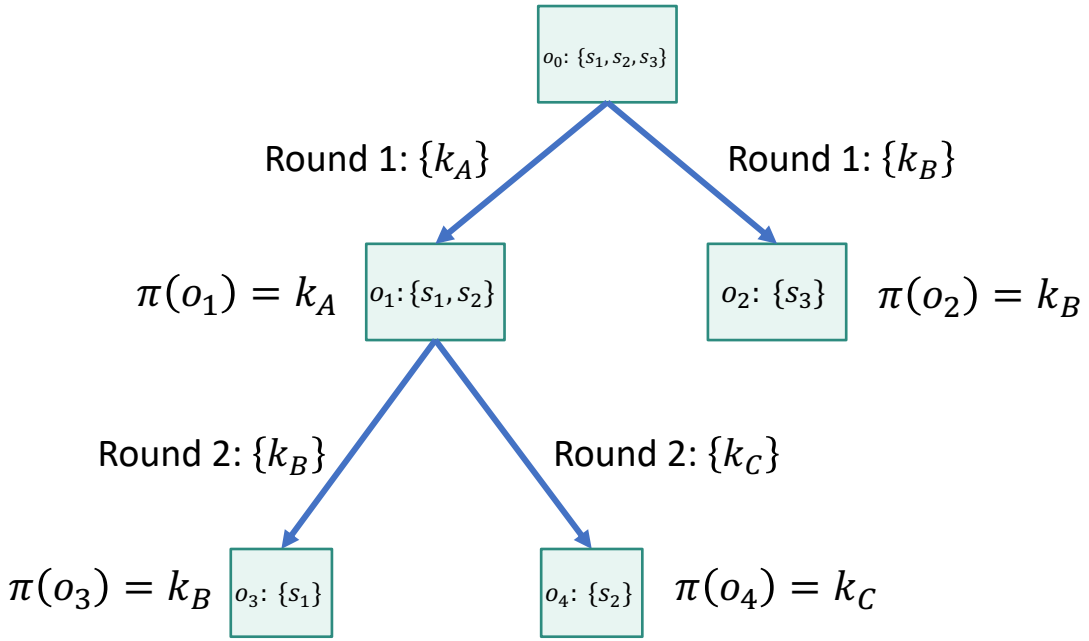


Edge (k, o) exists if key k can be played after observation o.

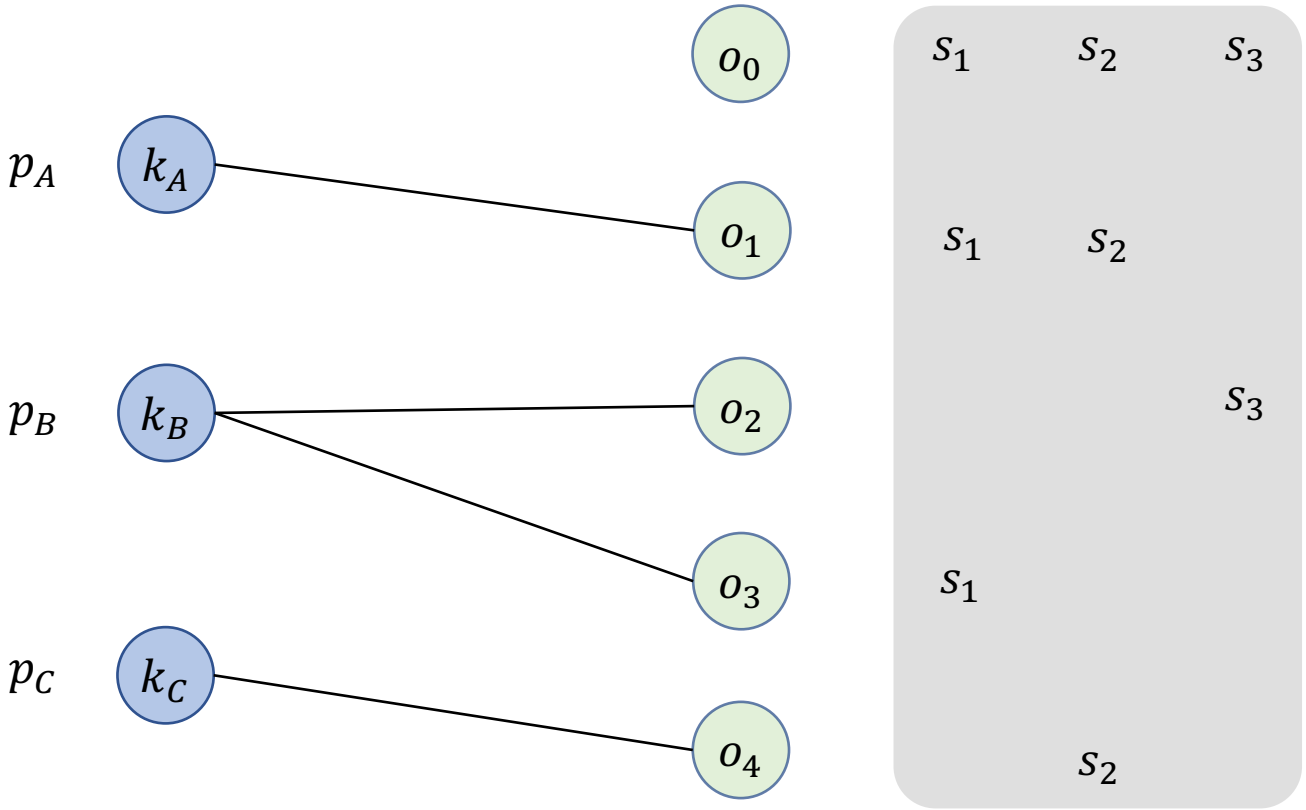
Fixed-Order Probabilistic Scenarios to MWLM



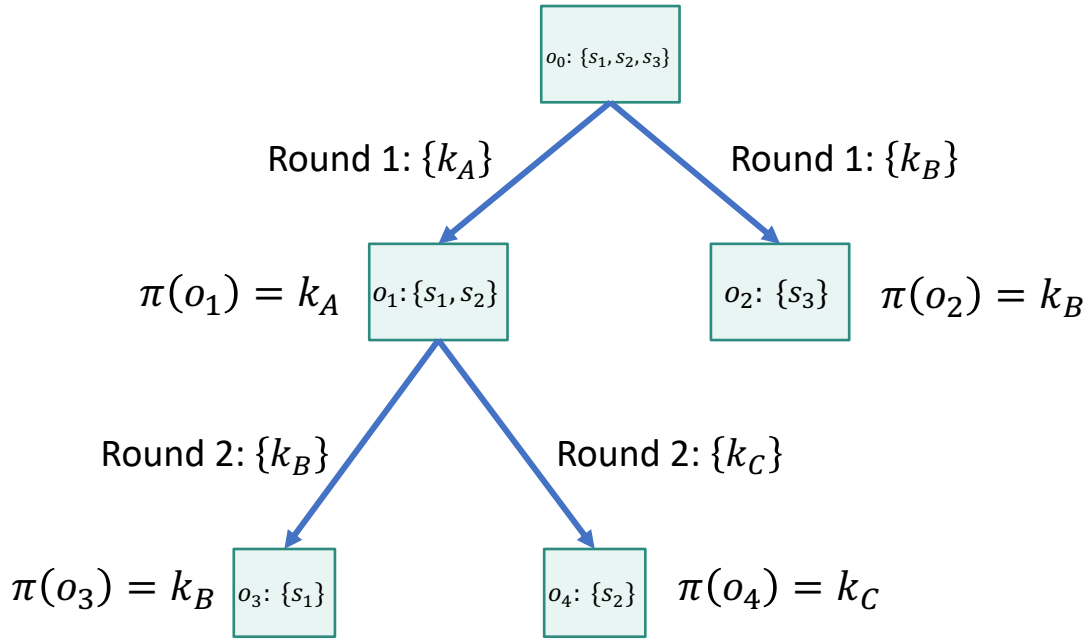
Edge (k, o) exists if key k can be played after observation o.



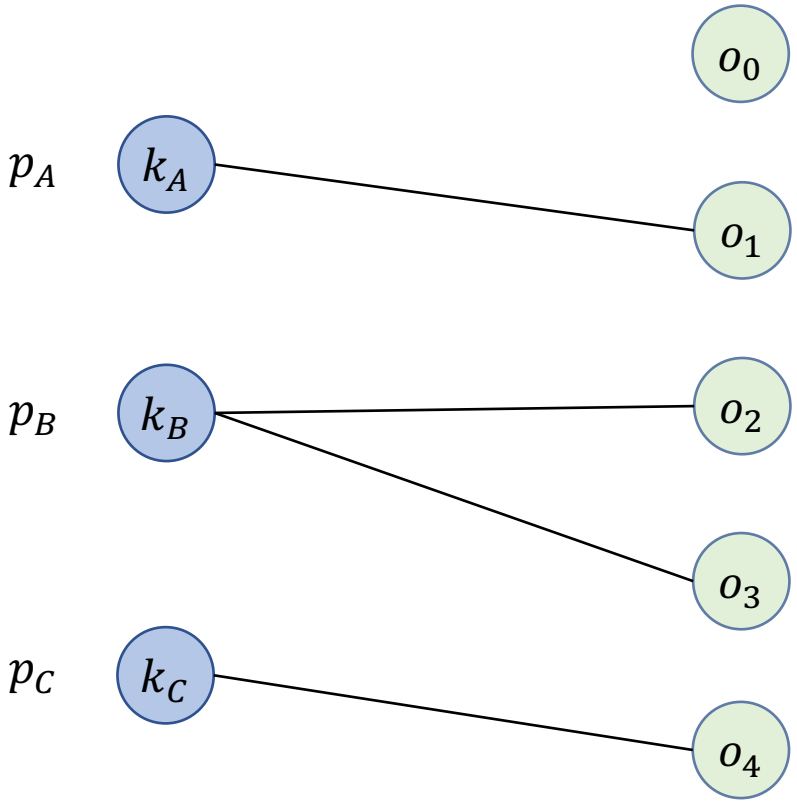
Fixed-Order Probabilistic Scenarios to MWLM

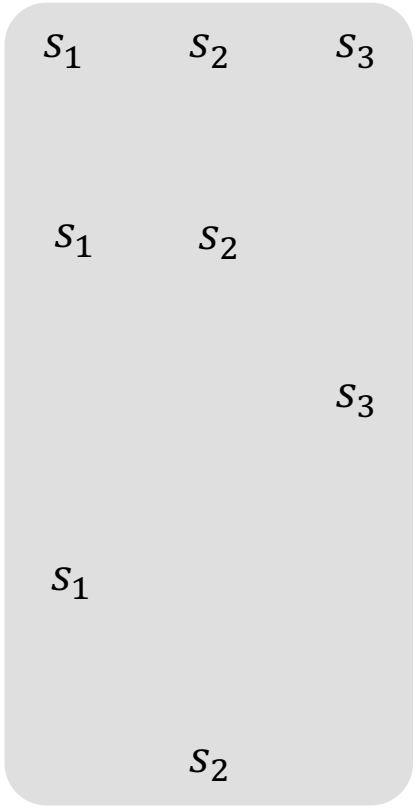


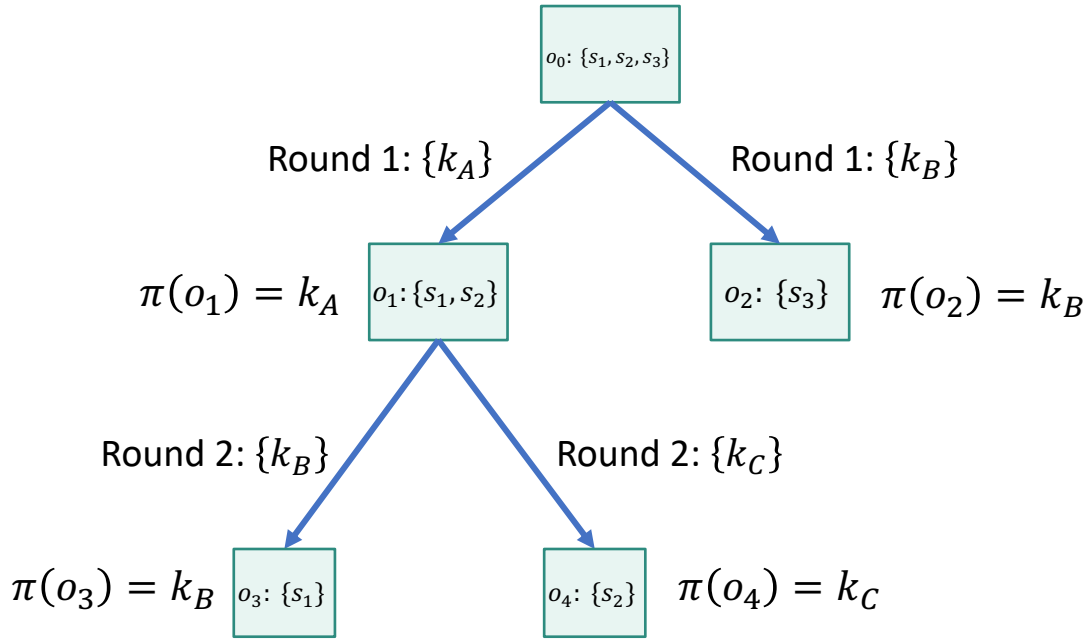
Edge (k, o) exists if key k can be played after observation o.



Fixed-Order Probabilistic Scenarios to MWLM



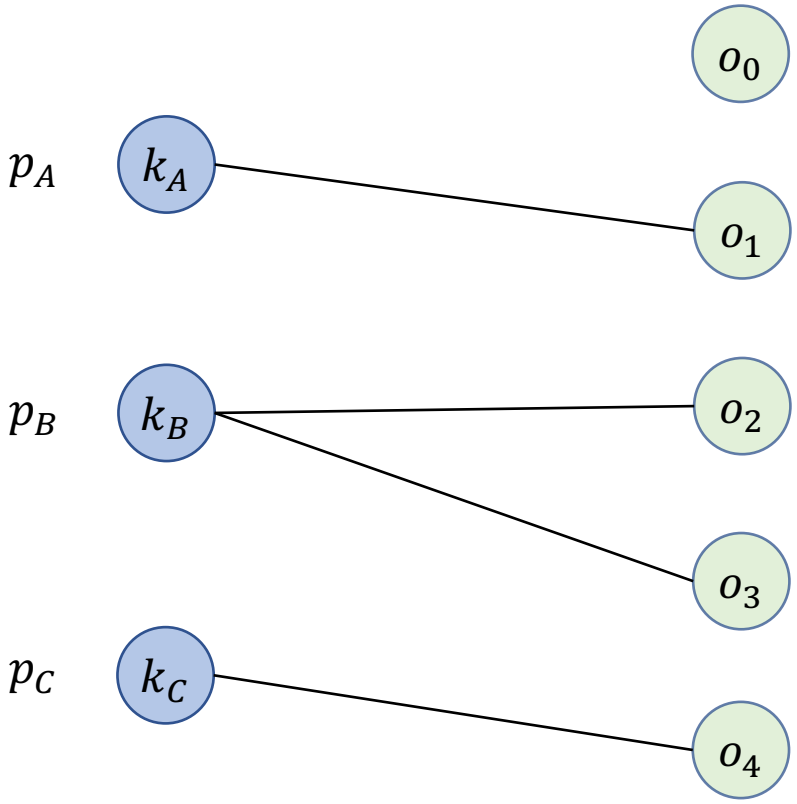


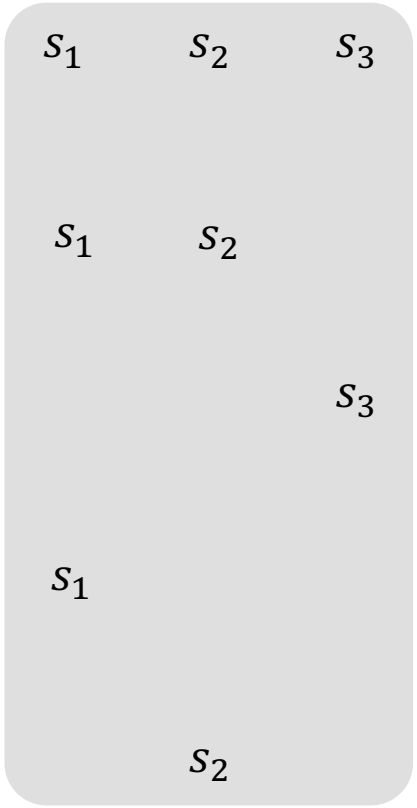


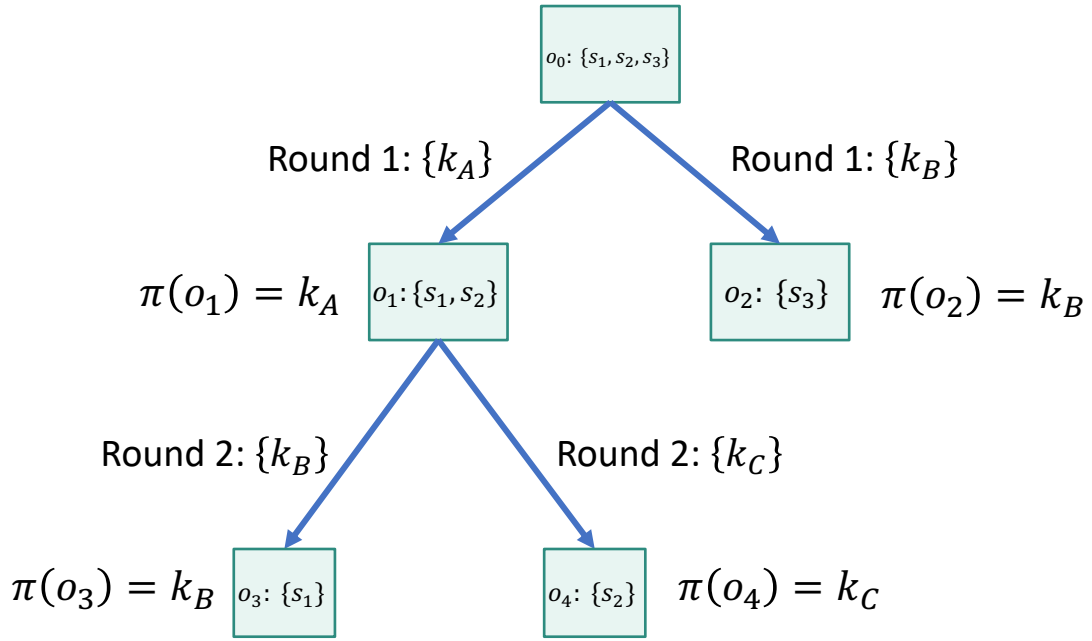
Edge (k, o) exists if key k can be played after observation o.

Weight of (k, o) is average reward of k over all consistent scenarios.

Fixed-Order Probabilistic Scenarios to MWLM





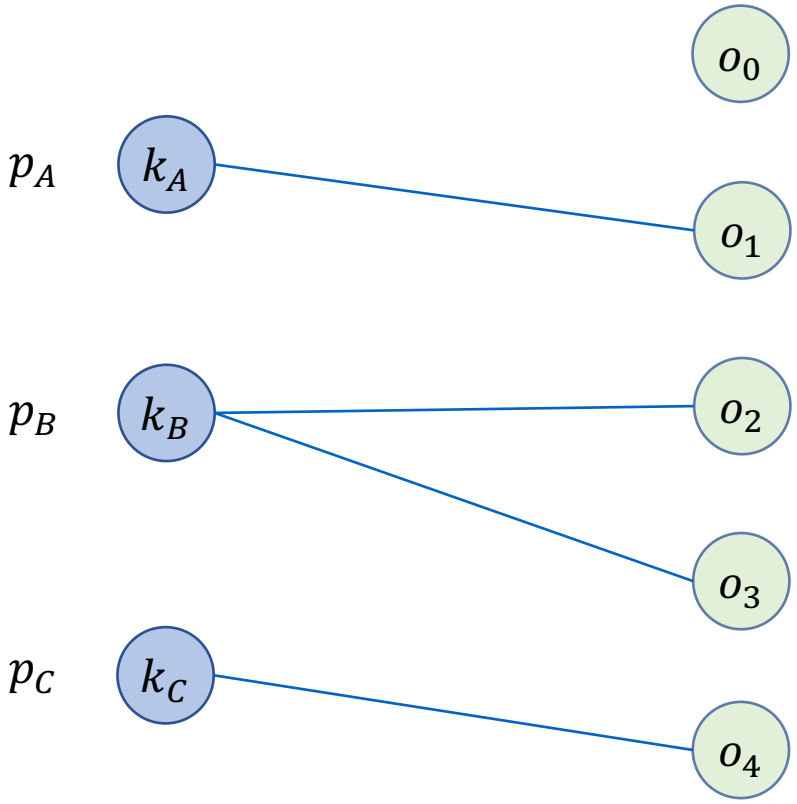


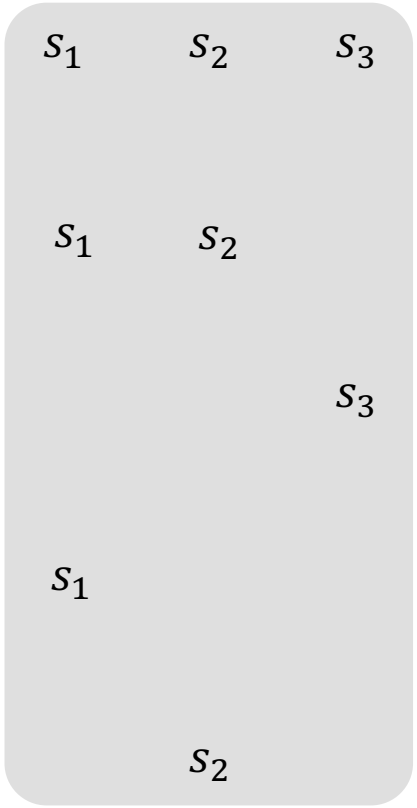
Edge (k, o) exists if key k can be played after observation o.

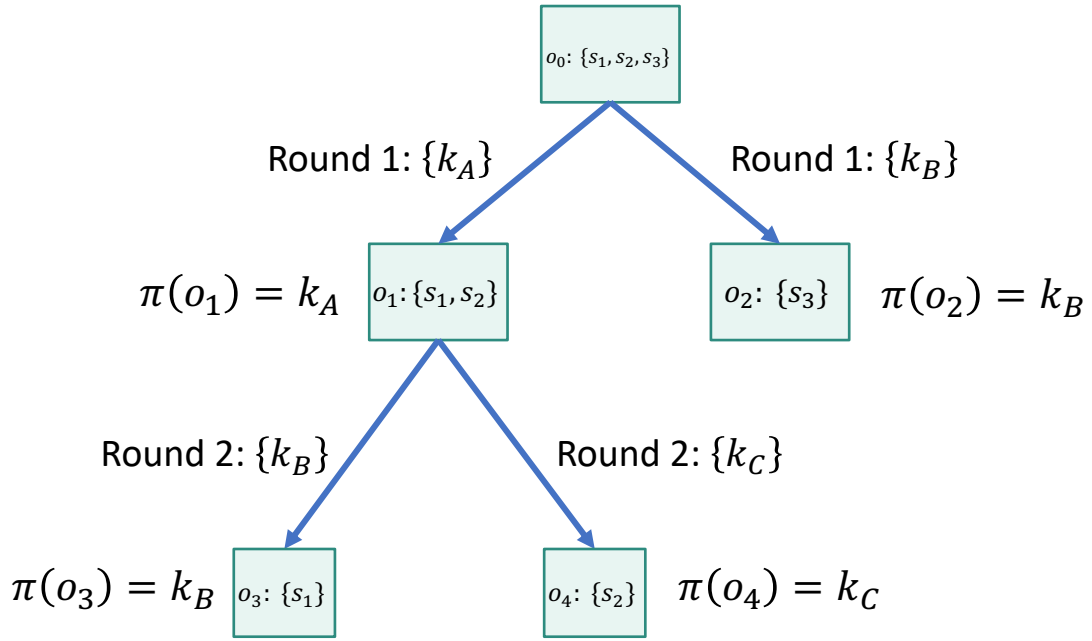
Weight of (k, o) is average reward of k over all consistent scenarios.

Correspondence between MWLM and *admissible* policies.

Fixed-Order Probabilistic Scenarios to MWLM







Edge (k, o) exists if key k can be played after observation o.

Weight of (k, o) is average reward of k over all consistent scenarios.

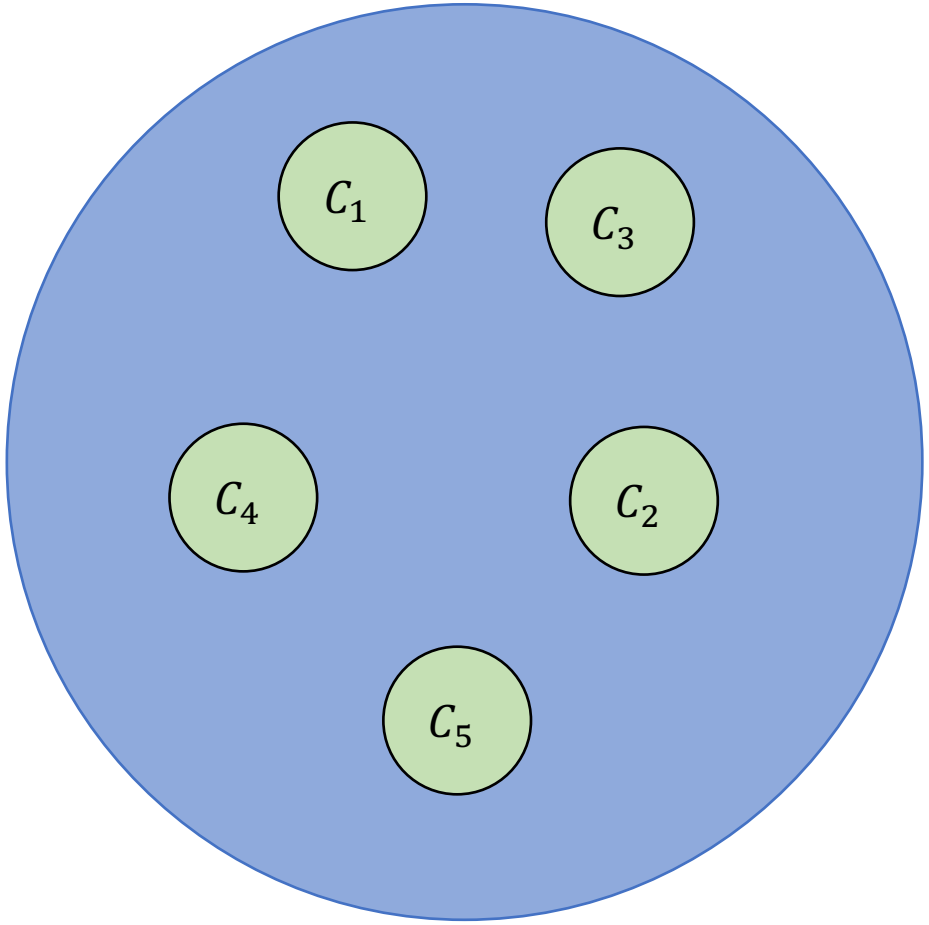
Correspondence between MWLM and *admissible* policies.

Order Selection

Question: What if we can select the order of keychains?

Setting: Optimizable Order

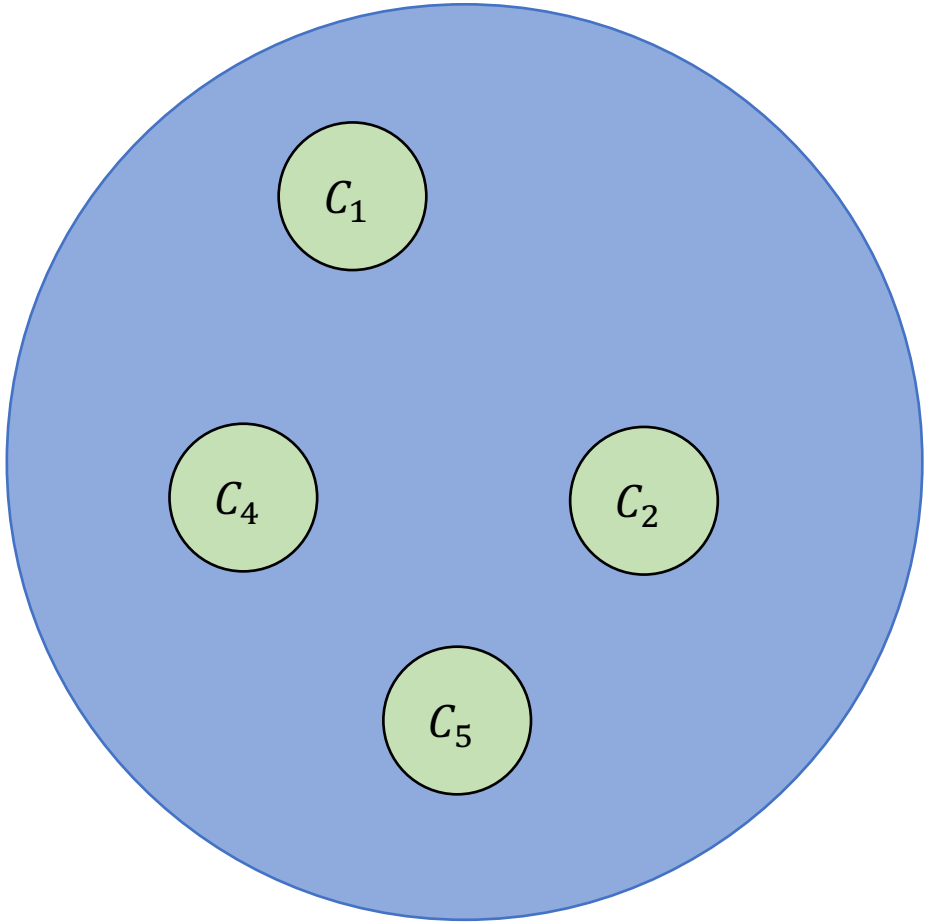
Order Selection



Key Played

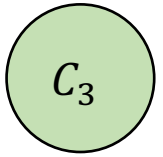
Chain

Order Selection

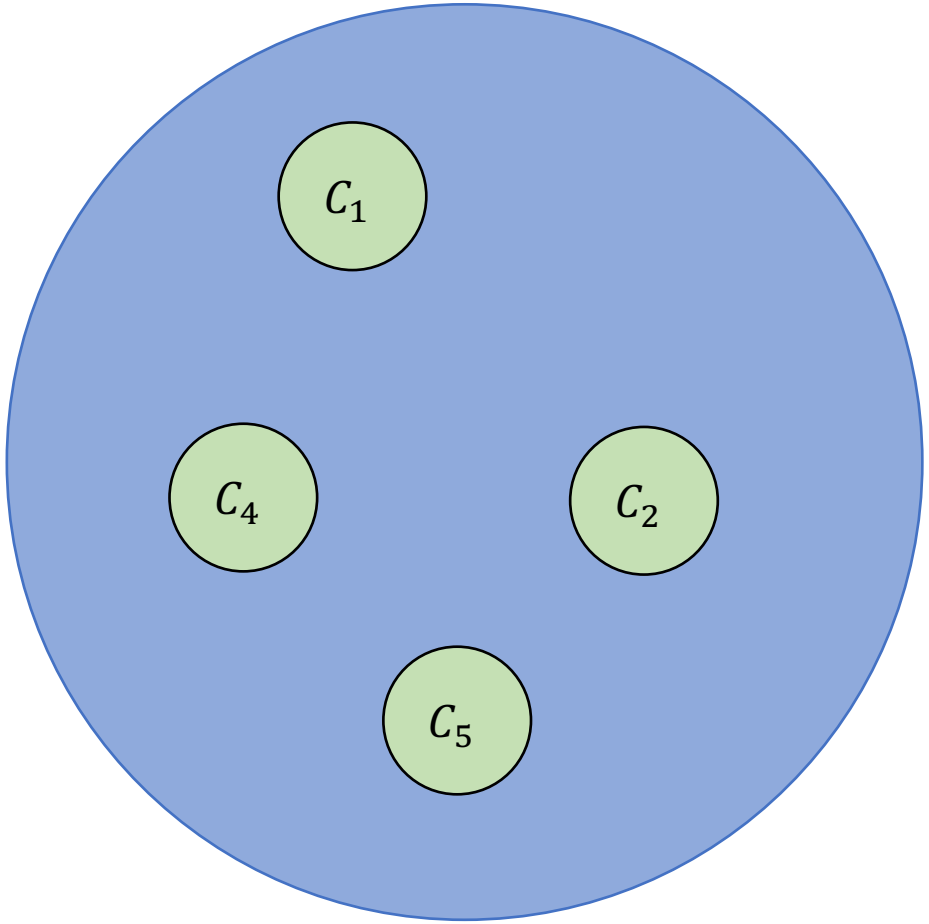


Key Played

Chain



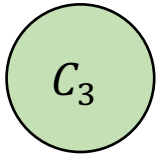
Order Selection



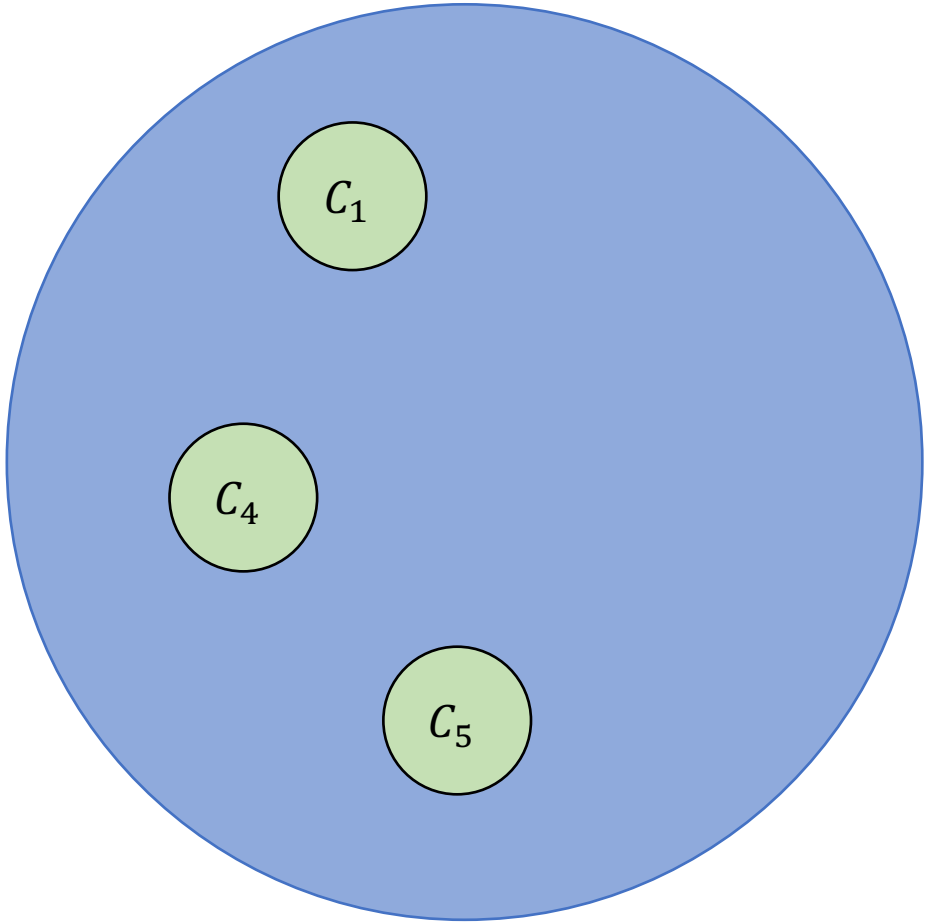
Key Played

k_B

Chain



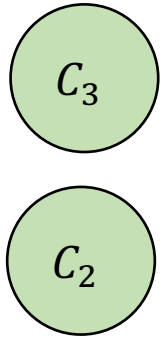
Order Selection



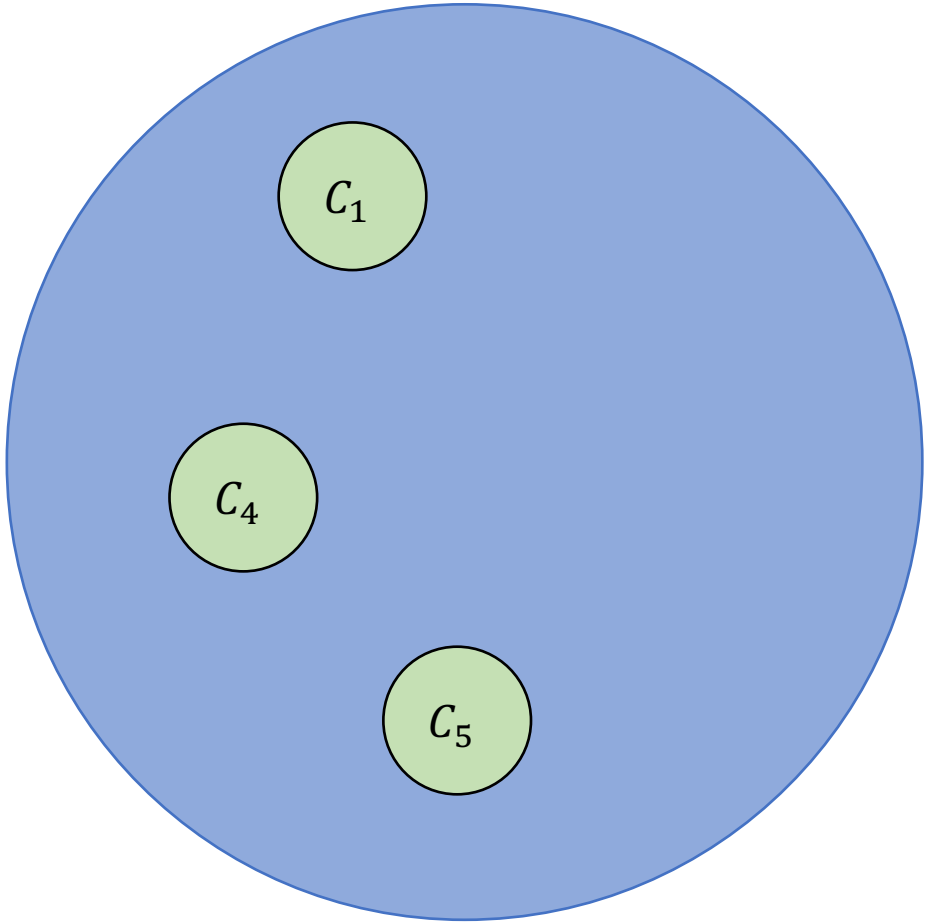
Key Played

k_B

Chain

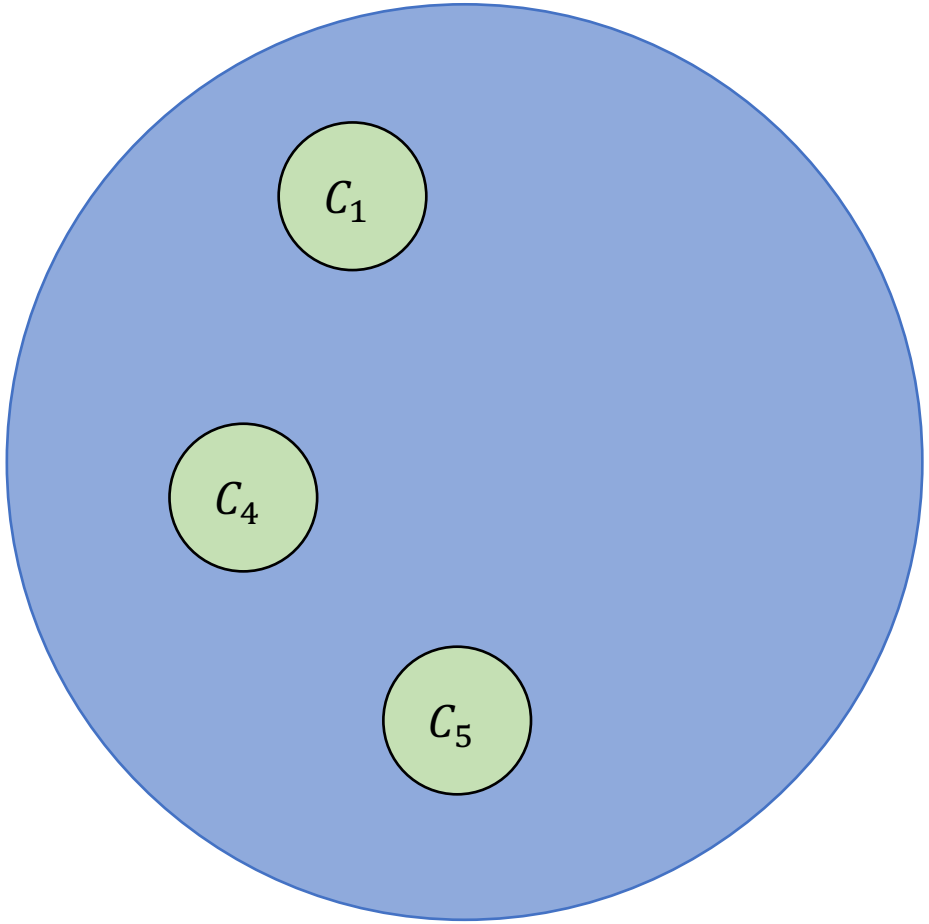


Order Selection



Key Played	Chain
k_B	C_3
k_A	C_2

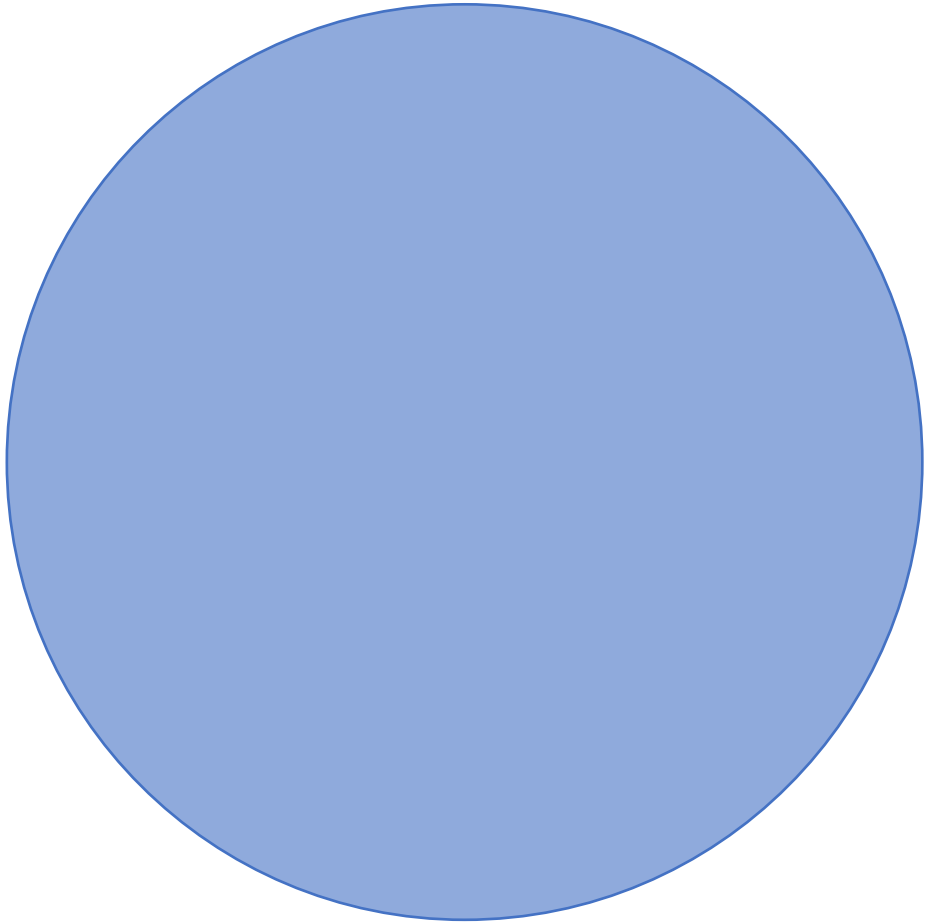
Order Selection



Key Played	Chain
k_B	C_3
k_A	C_2

Non-adaptive order is as good as adaptive order selection.

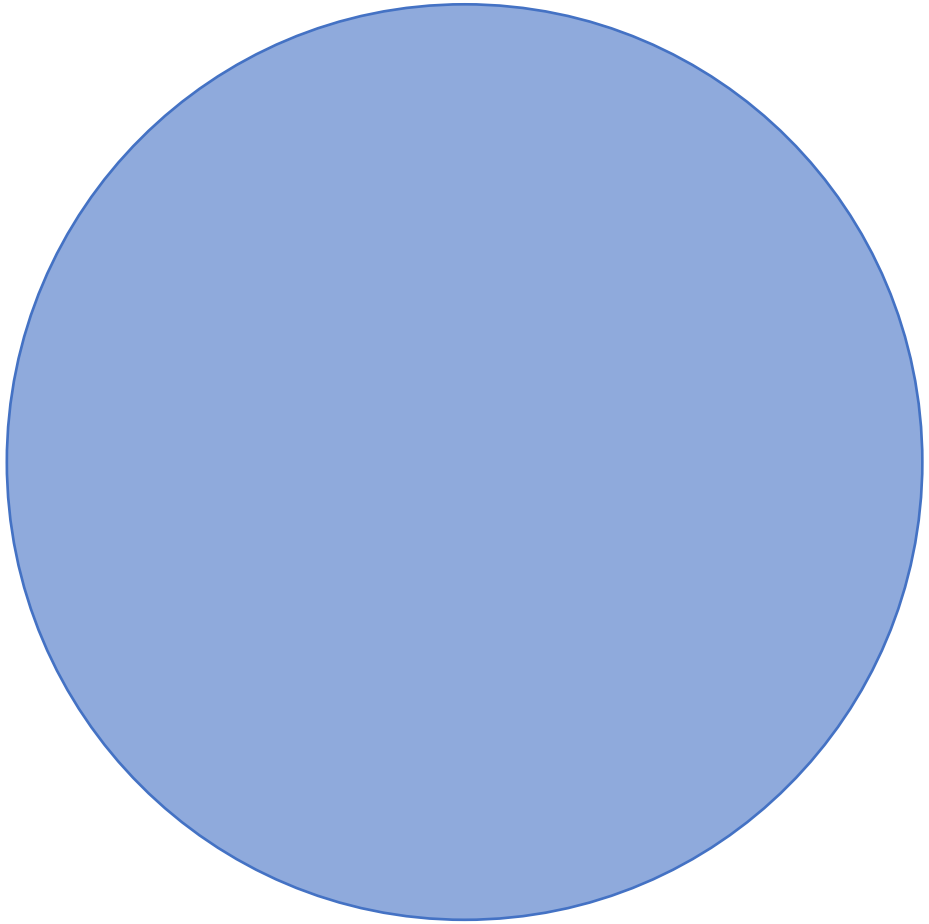
Order Selection



Key Played	Chain
k_B	C_3
k_A	C_2
	C_5
	C_4
	C_1

Non-adaptive order is as good as adaptive order selection.

Order Selection



Key Played	Chain
k_B	C_3
k_A	C_2
k_C	C_5
k_E	C_4
k_D	C_1

Non-adaptive order is as good as adaptive order selection.

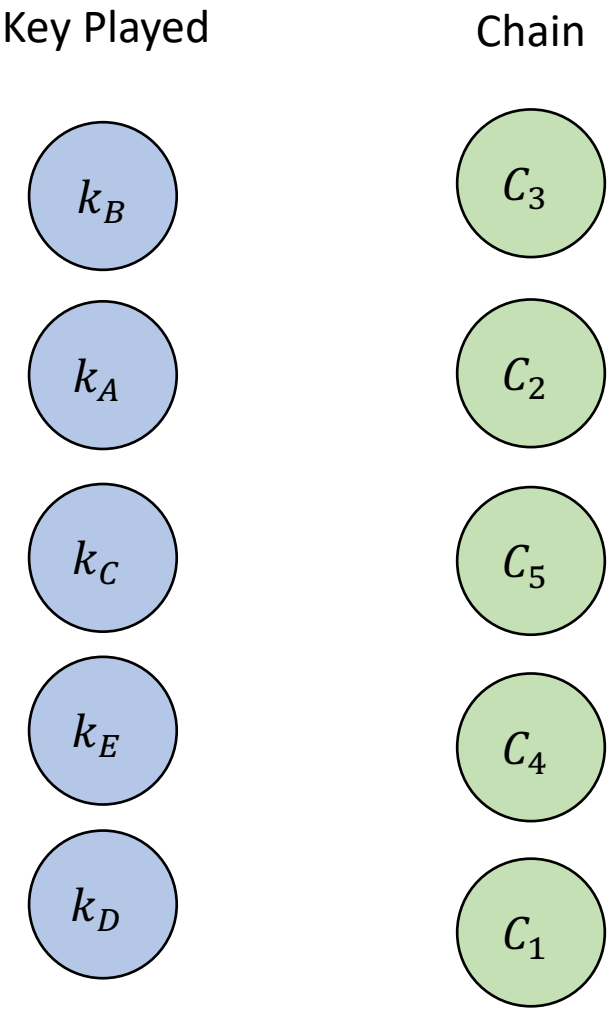
Order Selection as a Bipartite Graph Problem

Non-adaptive order is as good as adaptive order selection.

(k, c) ⇒ key k is on chain c. No fixed order ⇒ unweighted edges!

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.



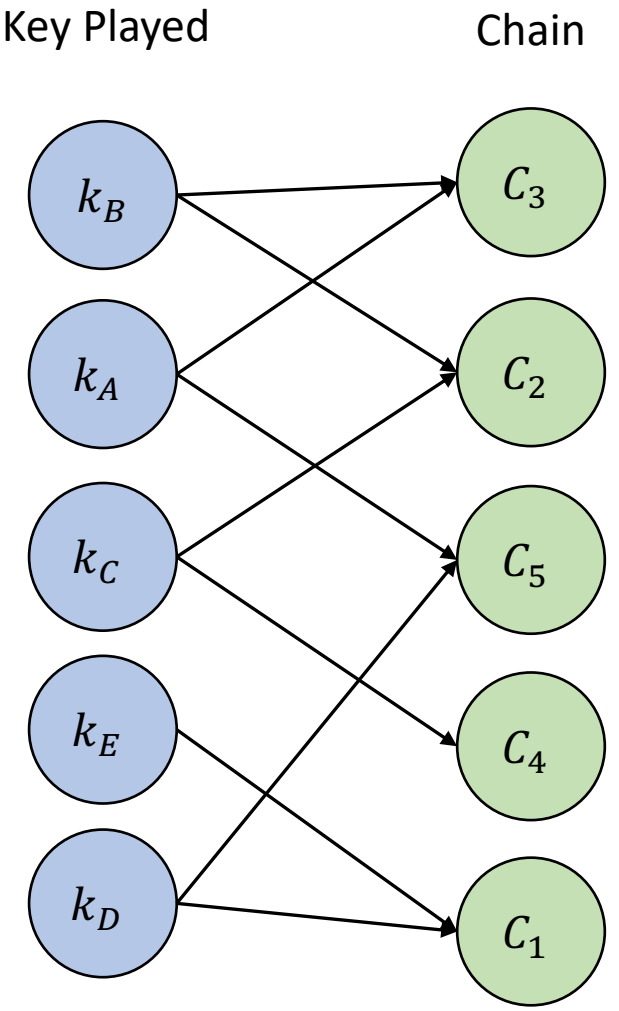
Order Selection as a Bipartite Graph Problem

Non-adaptive order is as good as adaptive order selection.

(k, c) ⇒ key k is on chain c. No fixed order ⇒ unweighted edges!

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

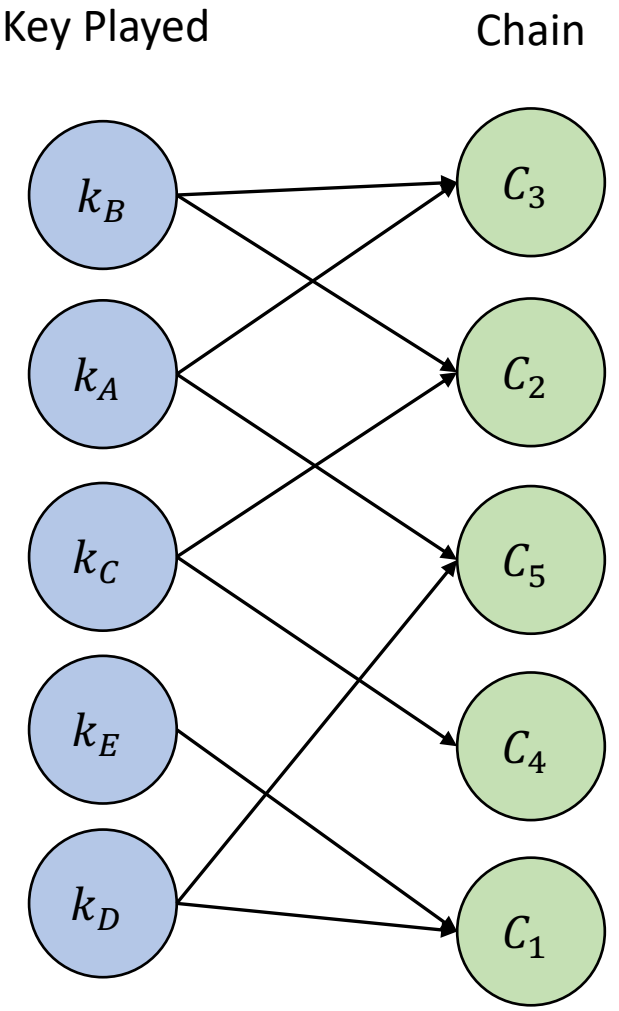
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.



Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

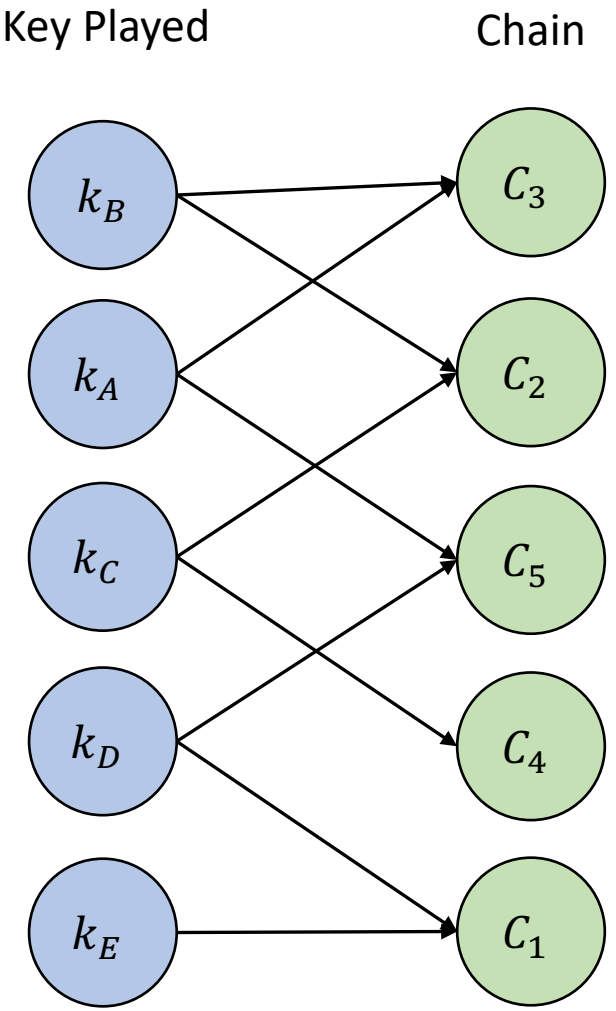
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.



Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

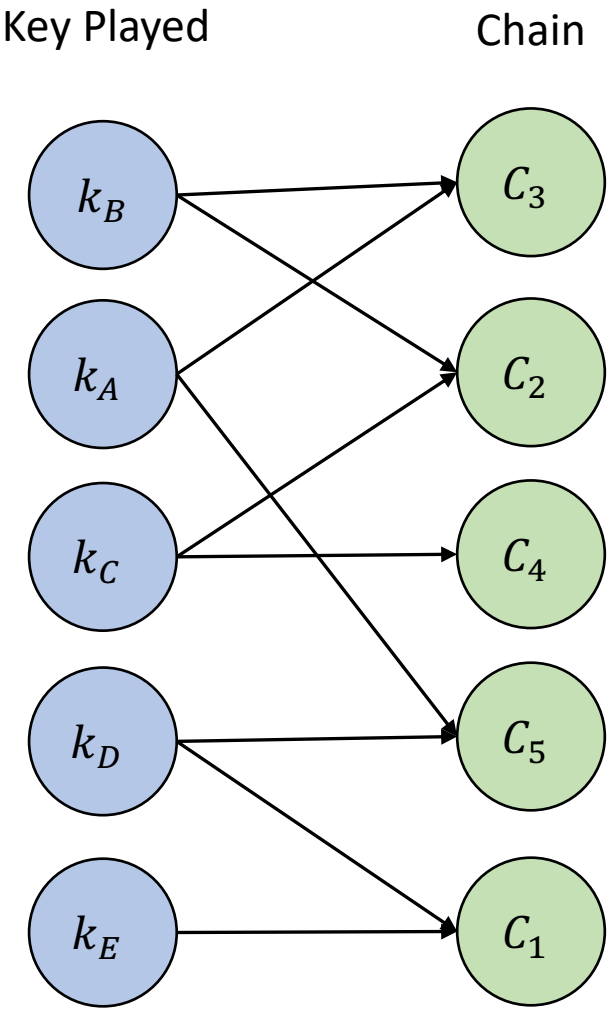
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.



Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

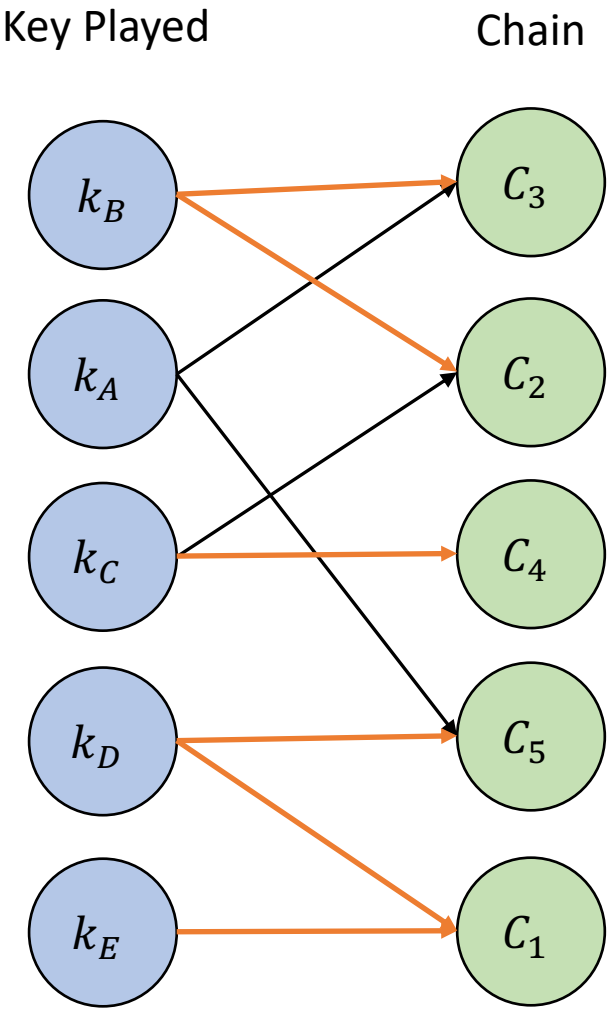
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.



Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

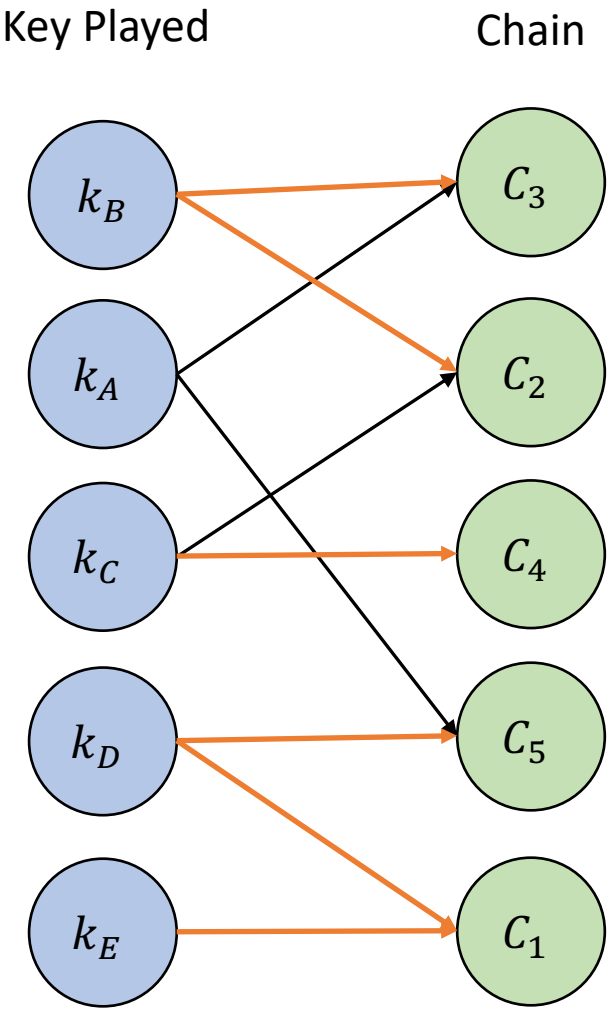
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.



Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.

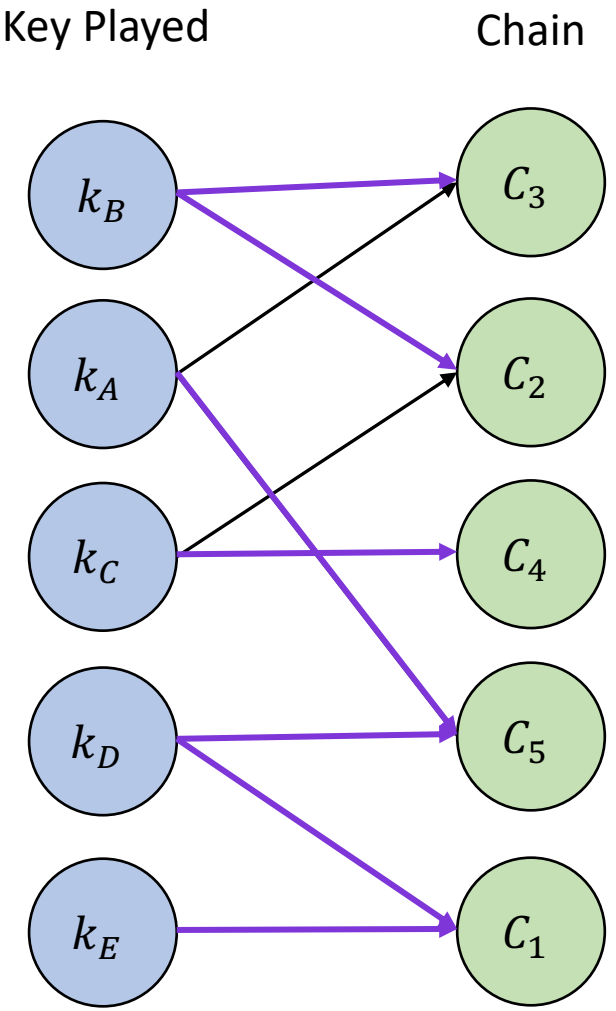
Downwards Facing Bipartite Permutation (DFBP): reorder keys and chains to maximize score.
Score of an ordering: number of non-upwards edges.



Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.

Downwards Facing Bipartite Permutation (DFBP): reorder keys and chains to maximize score.
Score of an ordering: number of non-upwards edges.

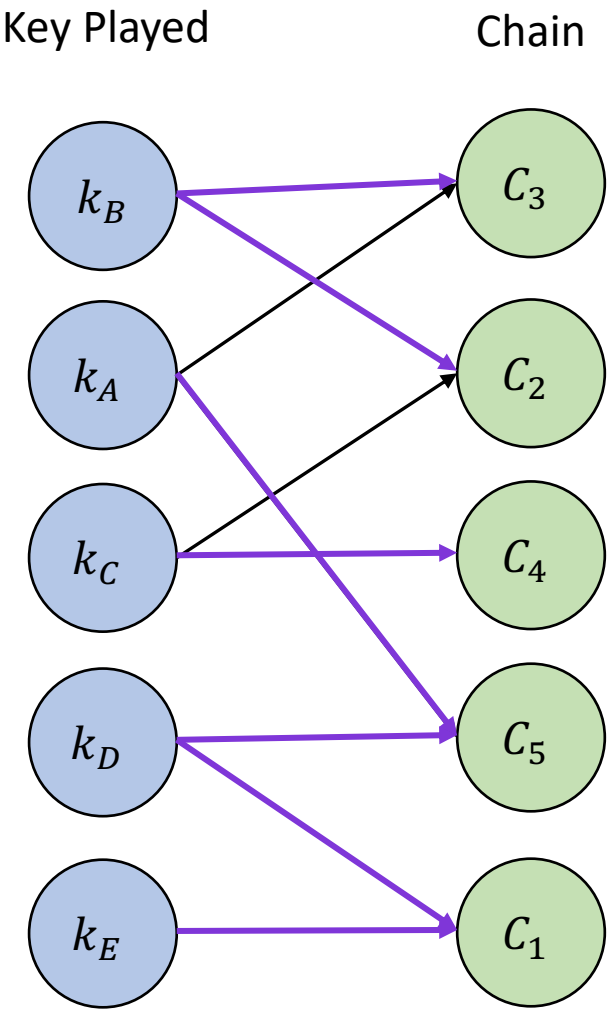


Order Selection as a Bipartite Graph Problem

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:
- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.

Downwards Facing Bipartite Permutation (DFBP): reorder keys and chains to maximize score.
Score of an ordering: number of non-upwards edges.

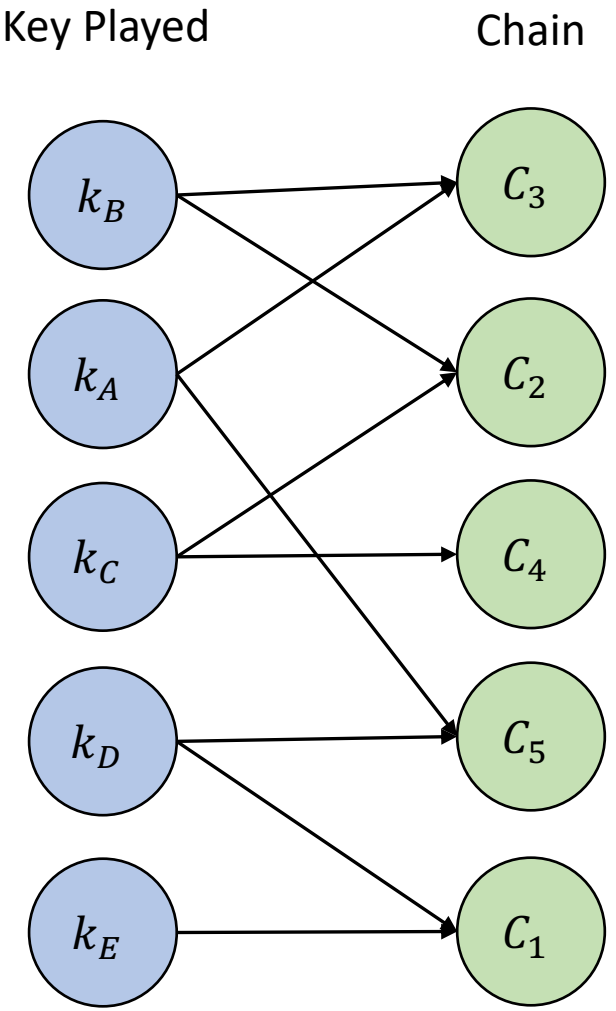
Theorem: Reduction from DFBP to Order Selection.
NP-hardness of DFBP ⇒ NP-hardness of Order Selection.



Approximating the Bayes Optimal Policy

Downwards Facing Bipartite Permutation (DFBP): reorder keys and chains to maximize score.
Score of an ordering: number of non-upwards edges.

Fix any ordering of keys and chains (σ_k, σ_C) and its reverse $(-\sigma_k, -\sigma_C)$. One of these two is a ½-approximation for the optimal DFBP ordering.

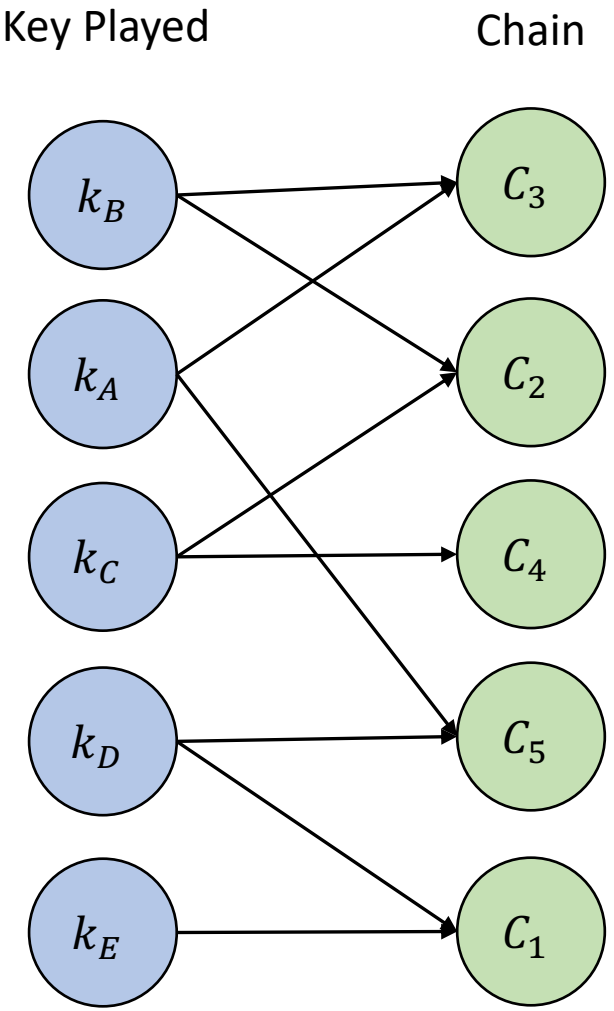


Approximating the Bayes Optimal Policy

Order Selection: reorder keys and chains to maximize score.
Score of an ordering:

- For each key, check it has a horizontal edge (corresponds to playing the key).
- Count number of non-upwards edges for played keys.

Fix any ordering of keys and chains (σ_k, σ_C) and its reverse $(-\sigma_k, -\sigma_C)$. Solving the Basic Keychain problem on one of these two is a ½-approximation for the optimal Order Selection policy.



Established Connections and Results

Keychain Variant	Related Bipartite Model	Useful Takeaways

Established Connections and Results

Keychain Variant	Related Bipartite Model	Useful Takeaways
Basic Keychain Problem	Maximum Weight Bipartite Matching	Bayes-optimal policy computable in polynomial time.

Established Connections and Results

Keychain Variant	Related Bipartite Model	Useful Takeaways
Basic Keychain Problem	Maximum Weight Bipartite Matching	Bayes-optimal policy computable in polynomial time.
Probabilistic Scenarios	Maximum Weight Laminar Matching	$\left(1-\frac{1}{e}\right)$-approximation to Bayes-optimal probabilistic scenarios policy.

Established Connections and Results

Keychain Variant	Related Bipartite Model	Useful Takeaways
Basic Keychain Problem	Maximum Weight Bipartite Matching	Bayes-optimal policy computable in polynomial time.
Probabilistic Scenarios	Maximum Weight Laminar Matching	$\left(1-\frac{1}{e}\right)$-approximation to Bayes-optimal probabilistic scenarios policy.
Order Selection	Downwards Facing Bipartite Permutation	DFBP ½-approximation idea gives ½-approximation to the Bayes-optimal order-selection policy.

Future Works

Future Works

- Succinct representation of exponentially many scenarios

Future Works

- Succinct representation of exponentially many scenarios
  - Uniform over all orderings

Future Works

- Succinct representation of exponentially many scenarios
  - Uniform over all orderings

- Formalize relationship between Order Selection setting and OCSPs

Future Works

- Succinct representation of exponentially many scenarios
  - Uniform over all orderings
- Formalize relationship between Order Selection setting and OCSPs
  - Is ½-approximation best-achievable?

Future Works

- Succinct representation of exponentially many scenarios
  - Uniform over all orderings
- Formalize relationship between Order Selection setting and OCSPs
  - Is ½-approximation best-achievable?
- What happens in settings with multiple correct keys?

Thank you!