

NICE: Robust Scheduling through Reinforcement Learning-Guided Integer Programming

USAF-MIT AI Accelerator

AAAI 22

Luke Kenworthy, Siddharth Nayak,
Christopher Chin and Hamsa Balakrishnan



**Massachusetts
Institute of
Technology**



Overview

- ➔ • **Crew Scheduling Problem**
- Optimization Baseline
- Robust Optimization
- Reinforcement Learning
- NICE



The Crew Scheduling Problem

- We are given a squadron (group) of pilots, and a collection of flights
- Each flight has multiple slots; each slot on a flight must be filled by a different pilot
- Each slot has qualification requirements that a pilot must meet
- Each pilot has days that they are unavailable
- Based on real-world needs; differs from industry

Overview

- Crew Scheduling Problem
- ➔ • **Optimization Baseline**
- Robust Optimization
- Reinforcement Learning
- NICE



Baseline IP Formulation

$i \in I$	The set of pilots
$f \in F$	The set of flights
$s \in S$	The set of all slots
$U_f \subset F$	Flights that conflict with flight f
$L_i \subset F$	Flights that conflict with pilot i 's leave
$S_f \subset S$	The set of slots belonging to flight f
$Q_i \subset S$	Slots that pilot i qualifies for

Baseline IP Formulation

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

such that:

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

Maximize number of
pilots getting assigned
to slots

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Baseline IP Formulation

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

such that:

$$X_{is} = 0 \quad \forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0 \quad \forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1 \quad \forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1 \quad \forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1 \quad \forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\} \quad \forall i \in I, s \in S$$

Pilots are only assigned to slots they are qualified for

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Pilots will never be assigned to flights that conflict with their leaves

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Do not assign pilots to multiple slots on same flight

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Every slot is filled by exactly one pilot

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Prevent assignment of
pilots to conflicting
flights

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$

$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Make decision variable
binary

Baseline IP Formulation

such that:

$$\max \sum_{i \in I} \sum_{s \in S} X_{is}$$

Due to the requirement of each slot having exactly one pilot, this will always give same objective value

$$X_{is} = 0$$

$$\forall i \in I, f \in L_i, s \in S_f$$

$$X_{is} = 0$$

$$\forall i \in I, s \in S \setminus Q_i$$

$$\sum_{s \in S_f} X_{is} \leq 1$$

$$\forall i \in I, f \in F$$

$$\sum_{i \in I} X_{is} = 1$$

$$\forall s \in S$$

$$\sum_{s \in S_f} X_{is} + \sum_{s' \in S_{f'}} X_{is'} \leq 1$$

$$\forall i \in I, f \in F, f' \in U_f$$




$$X_{is} \in \{0, 1\}$$

$$\forall i \in I, s \in S$$

Baseline IP Formulation

Slots

Pilots

	\square	...	\circ	...	\triangle
	X_{11}		X_{12}		X_{13}
...					
	X_{21}		X_{22}		X_{23}
...					
	X_{31}		X_{32}		X_{33}

Overview

- Crew Scheduling Problem
- Optimization Baseline
- ➔ • **Robust Optimization**
- Reinforcement Learning
- NICE



Buffer Formulation

- To optimize schedules for robustness:
 - Increase the amount of buffer time between flights
- Identify all flight pairings that would create a buffer less than or equal to some max threshold.

Create a $\{0, 1\}$ decision variable $B_{iff'}$, \forall pilots $i \in I$, flights $f, f' \in F \times F$

Add constraints to ensure that $B_{iff'}$ is 1 if and only if the following conditions are met:

1 : $f \neq f'$

2 : Pilot i qualifies for at least one slot in both f and f'

3 : f and f' have a buffer between 0 and T_{buffer}

4 : Pilot i is assigned to both f and f' , without any flights in between

Buffer Formulation

- To optimize schedules for robustness:
 - Increase the amount of buffer time between flights
- Identify all flight pairings that would create a buffer less than or equal to some max threshold.

Create a $\{0, 1\}$ decision variable $B_{iff'}$, \forall pilots $i \in I$, flights $f, f' \in F \times F$
Add constraints to ensure that $B_{iff'}$ is 1 if and only if the following conditions are met:

- 1 : $f \neq f'$
- 2 : Pilot i qualifies for at least one slot in both f and f'
- 3 : f and f' have a buffer between 0 and T_{buffer}
- 4 : Pilot i is assigned to both f and f' , without any flights in between

Basically, B is 1 if a pilot is assigned to two successive flights that are close together, and is qualified for at least one slot on both flights.

Buffer Formulation

- Change the objective function to maximize buffers of successive flights pilots are assigned to:

$$\max \sum_{i \in I} \sum_{f, f' \in F \times F} b_{iff'} B_{iff'}$$

Where $b_{iff'} \in [-1, 0)$ is a buffer penalty which penalises for having smaller buffers

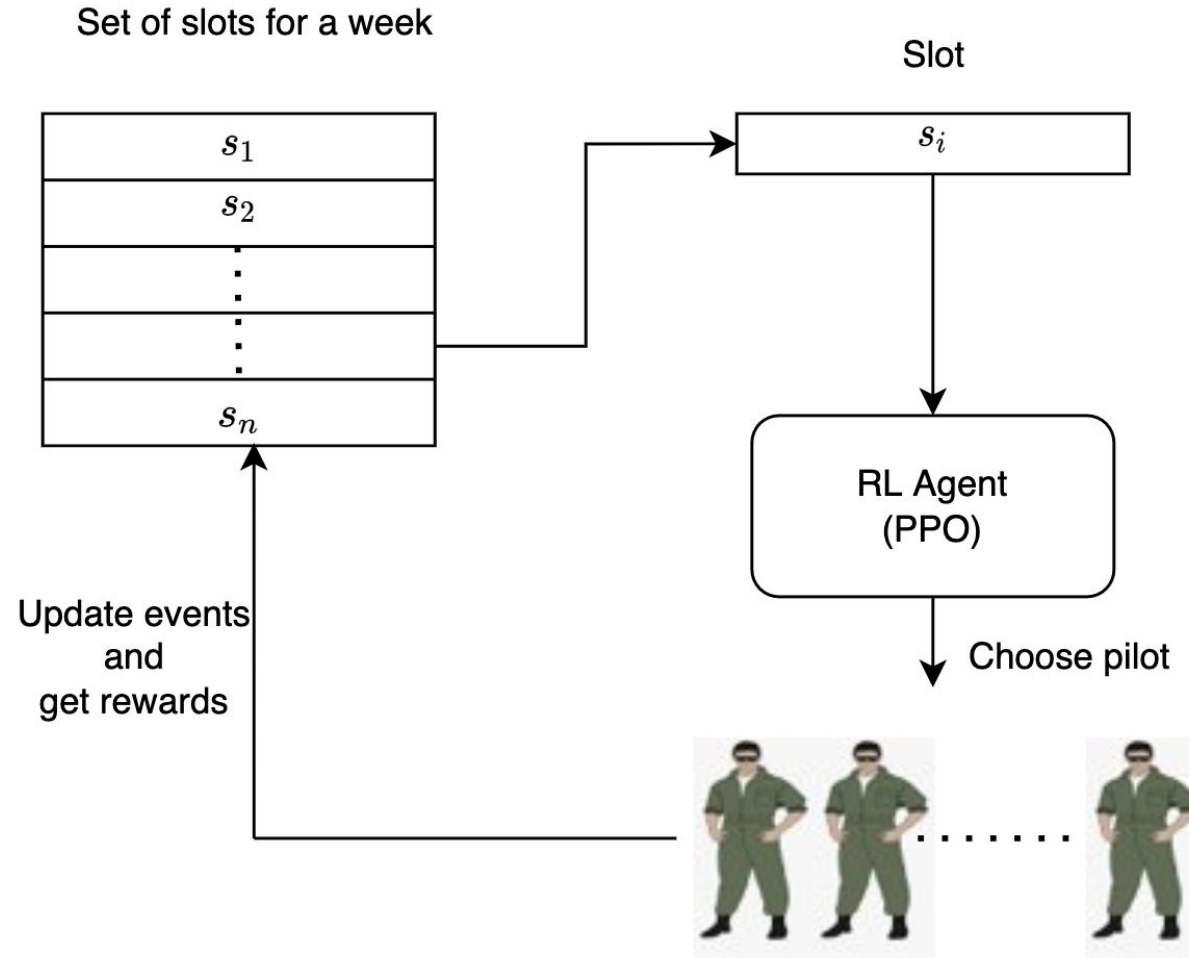
- **Problem:** This formulation is computationally intensive, quickly becoming intractable.

Overview

- Crew Scheduling Problem
- Optimization Baseline
- Robust Optimization
- ➔ • **Reinforcement Learning**
- NICE



Reinforcement Learning for Scheduling



DES for the RL Agent

- **State Space:**

- Event information: type, minimum qualification required, duration, start-end dates, pilots assigned
- Valid Pilots for the event
- Training requirements for pilots

- **Action Space:**

- Choose one of the pilots from the probability distribution predicted by the agent

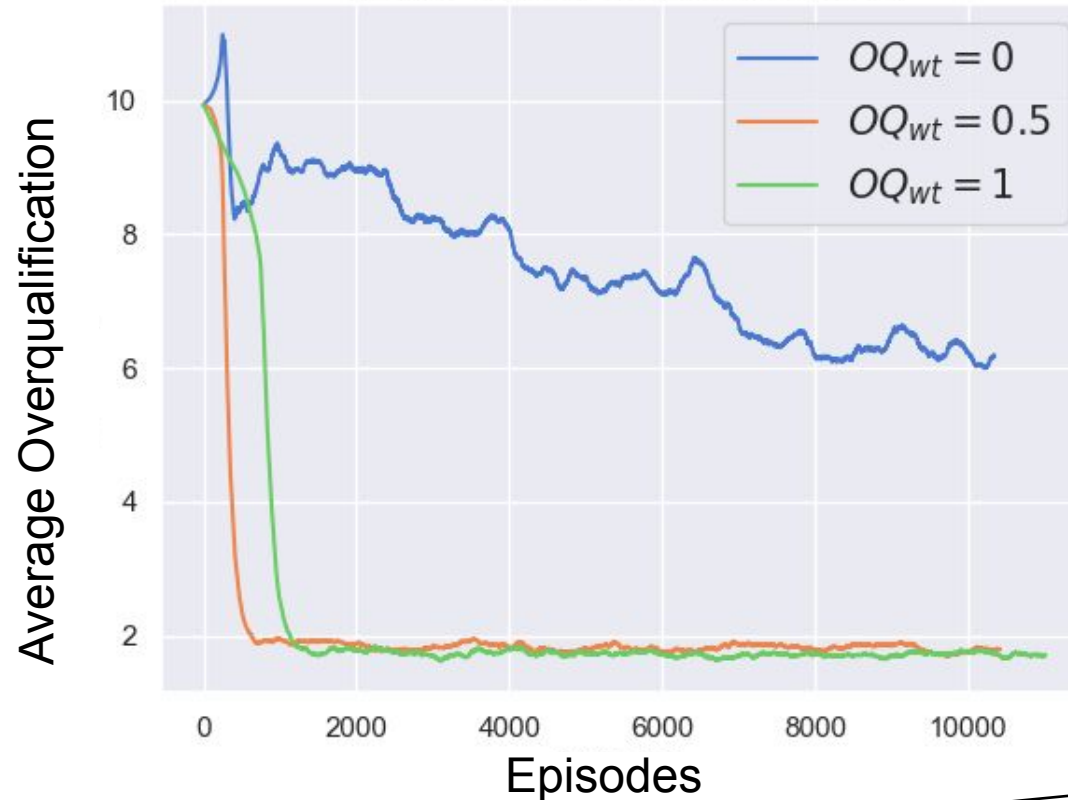
- **Rewards:**

- Must give rewards locally to incentivize RL agent to build optimal schedule

- **Time corresponds to slots**

Example: Minimizing Overqualification

Average Overqualification per Slot



$$r_t = OQ_{wt} \cdot \tanh(OQ_{cutoff} - OQ) + (1 - OQ_{wt})$$

Penalty for overqualification

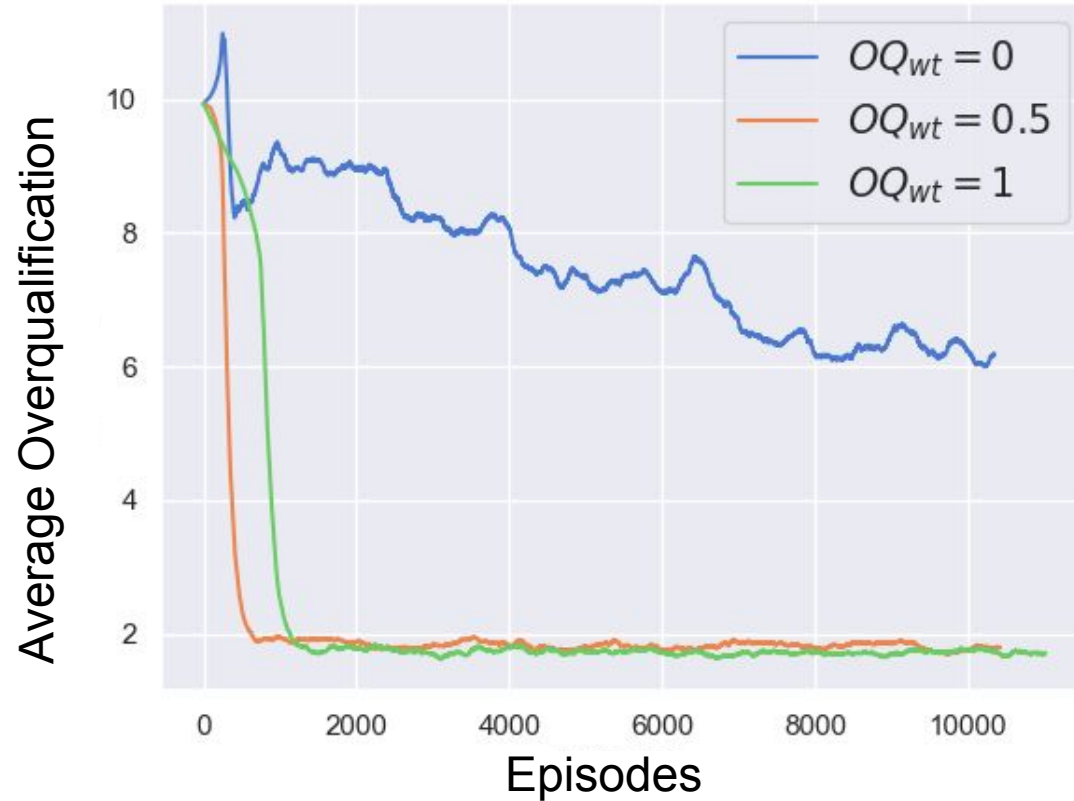
Reward for placing a pilot

OQ = Rank of pilot assigned - Minimum rank required for event slot

OQ_{cutoff} = Allowance for overqualification (= 5 for all experiments)

Example: Minimizing Overqualification

Average Overqualification per Slot



- **Problem 1:** Works well but, with perfect information (i.e. without uncertainty), IP performs better in a reasonable amount of time.
- **Problem 2:** Does not work as well for robust scheduling. The RL scheduler schedules greedily, potentially backing it into a bad solution based on early assignments.

$$r_t = OQ_{wt} \cdot \tanh(OQ_{cutoff} - OQ) + (1 - OQ_{wt})$$

OQ = Rank of pilot assigned - Minimum rank required for event slot

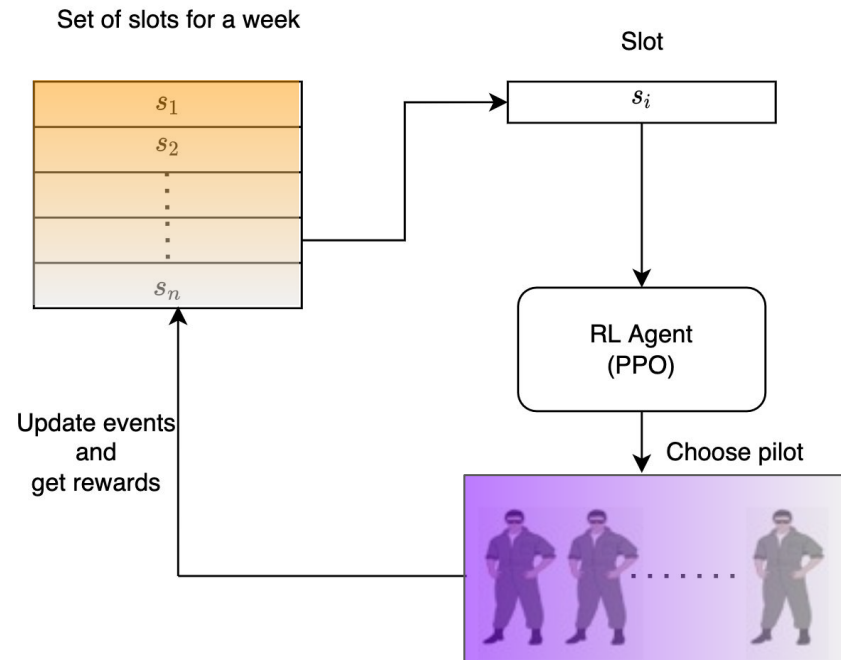
OQ_{cutoff} = Allowance for overqualification (= 5 for all experiments)

Overview

- Crew Scheduling Problem
- Optimization Baseline
- Robust Optimization
- Reinforcement Learning
- ➔ • **NICE**



Combining Reinforcement Learning with IP



Isomorphism

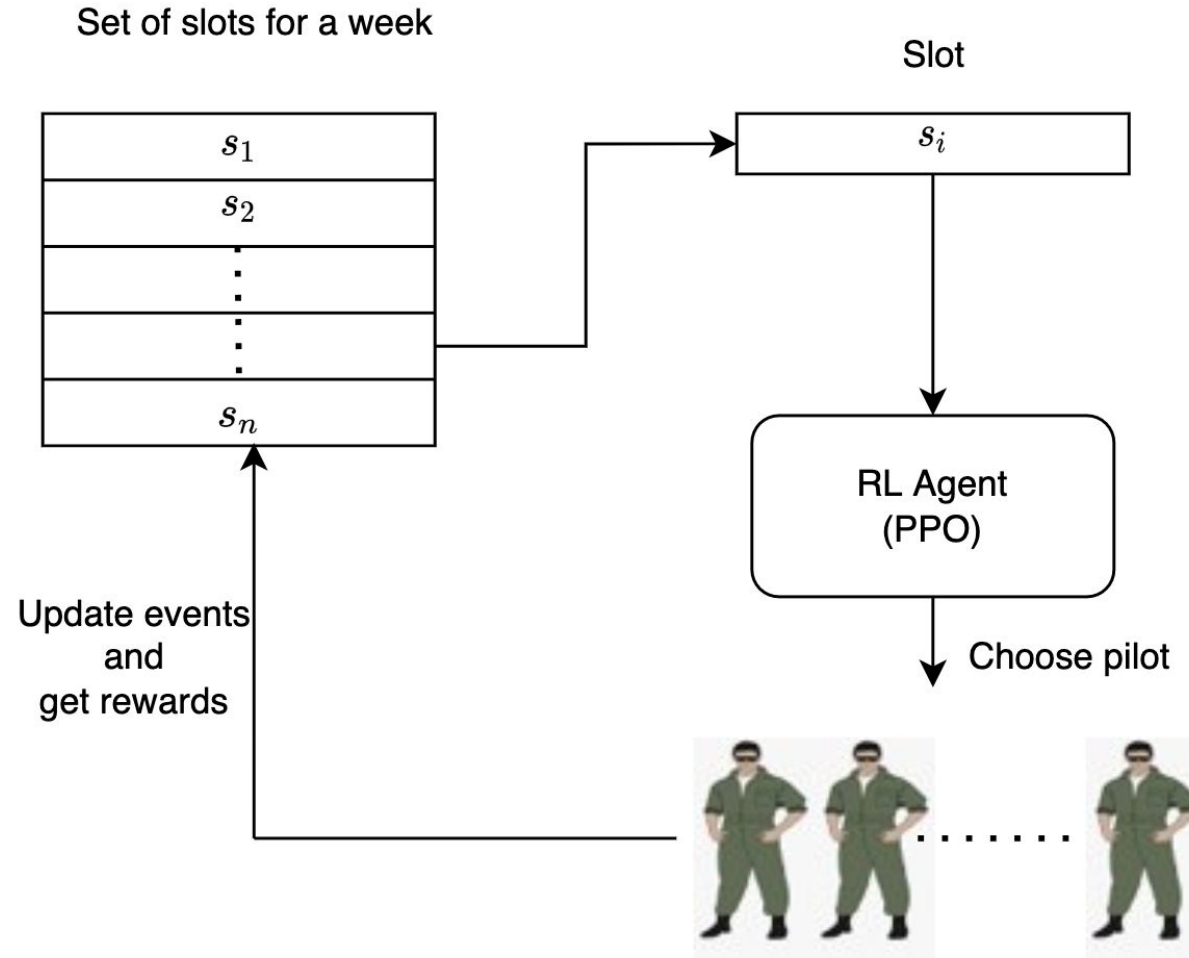


Pilots

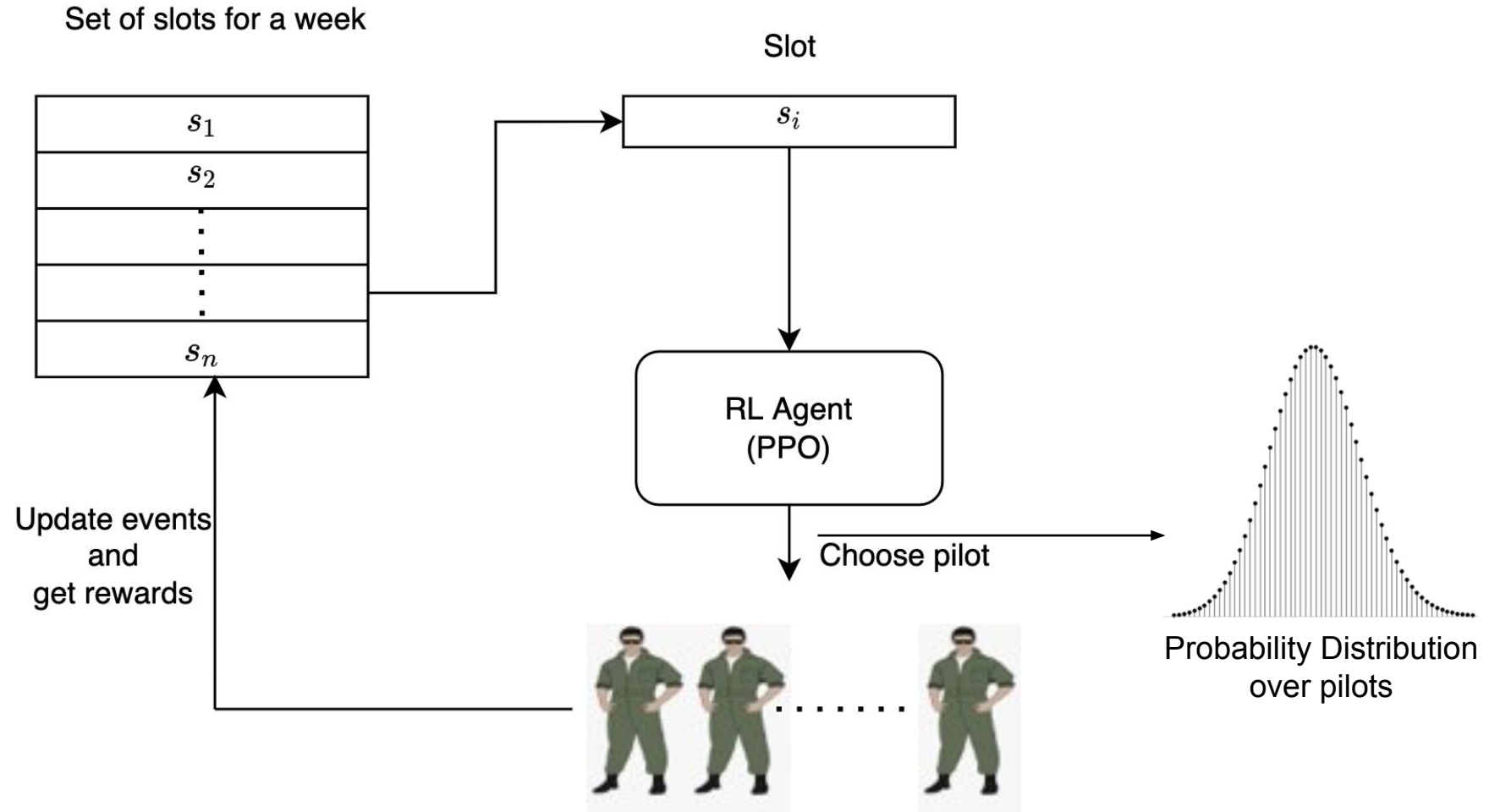
Slots

	□	...	○	...	△
✈	X_{11}		X_{12}		X_{13}
...					
✈	X_{21}		X_{22}		X_{23}
...					
✈	X_{31}		X_{32}		X_{33}

Combining Reinforcement Learning with IP



Combining Reinforcement Learning with IP

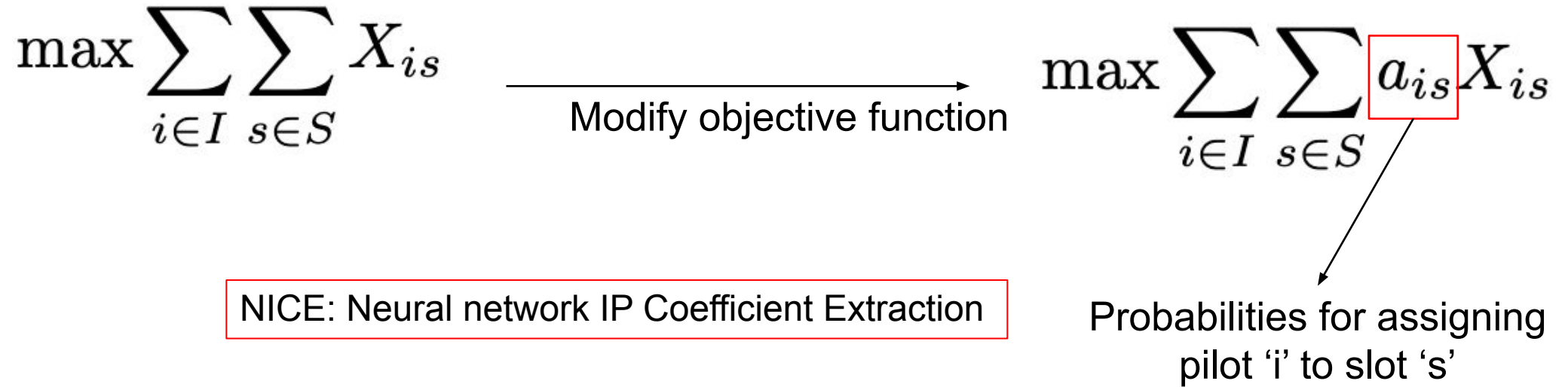


Combining Reinforcement Learning with IP

$$\max \sum_{i \in I} \sum_{s \in S} X_{is} \xrightarrow{\text{Modify objective function}} \max \sum_{i \in I} \sum_{s \in S} a_{is} X_{is}$$

Probabilities for assigning pilot 'i' to slot 's'

Combining Reinforcement Learning with IP



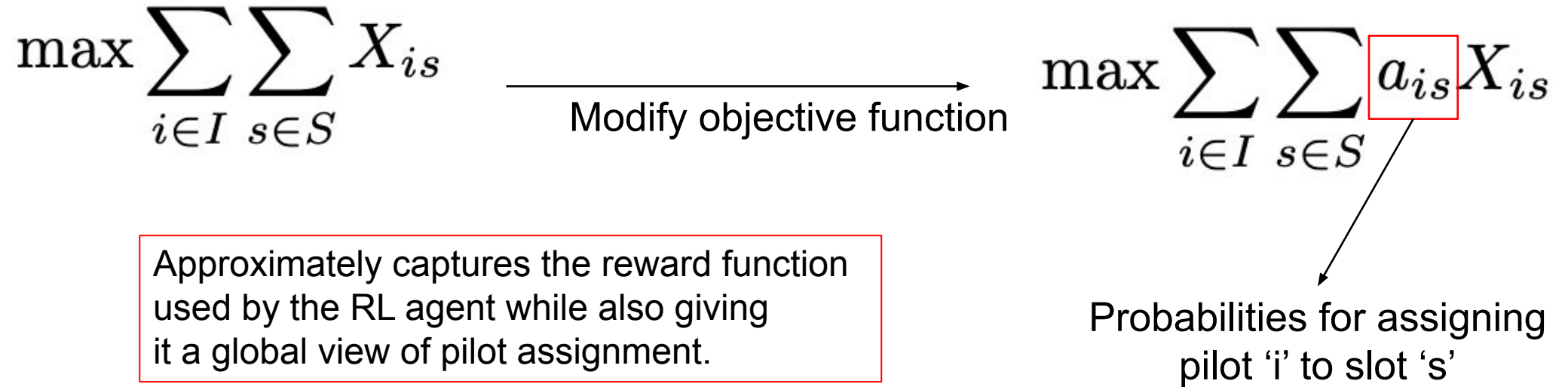
Combining Reinforcement Learning with IP

$$\max \sum_{i \in I} \sum_{s \in S} X_{is} \xrightarrow{\text{Modify objective function}} \max \sum_{i \in I} \sum_{s \in S} a_{is} X_{is}$$

Incentivizes the agent to pick the pilot with the highest probability possible at each slot, only selecting pilots with lower probabilities if constraints forbid such an assignment.

Probabilities for assigning pilot 'i' to slot 's'

Combining Reinforcement Learning with IP



Extracting Probability Weights

- The actions taken at each state of the RL scheduling process can impact the pilot probability vector at later states.
- Simply running the RL scheduling process as normal on a given problem instance while extracting the probability vector at each slot could cause the specific actions taken to bias the probability values, diminishing the advantage of the IP scheduler's global outlook.
- To tackle this, we use two different approaches to extract the probability weights:
 - Monte Carlo Approach
 - Blank Slate Approach

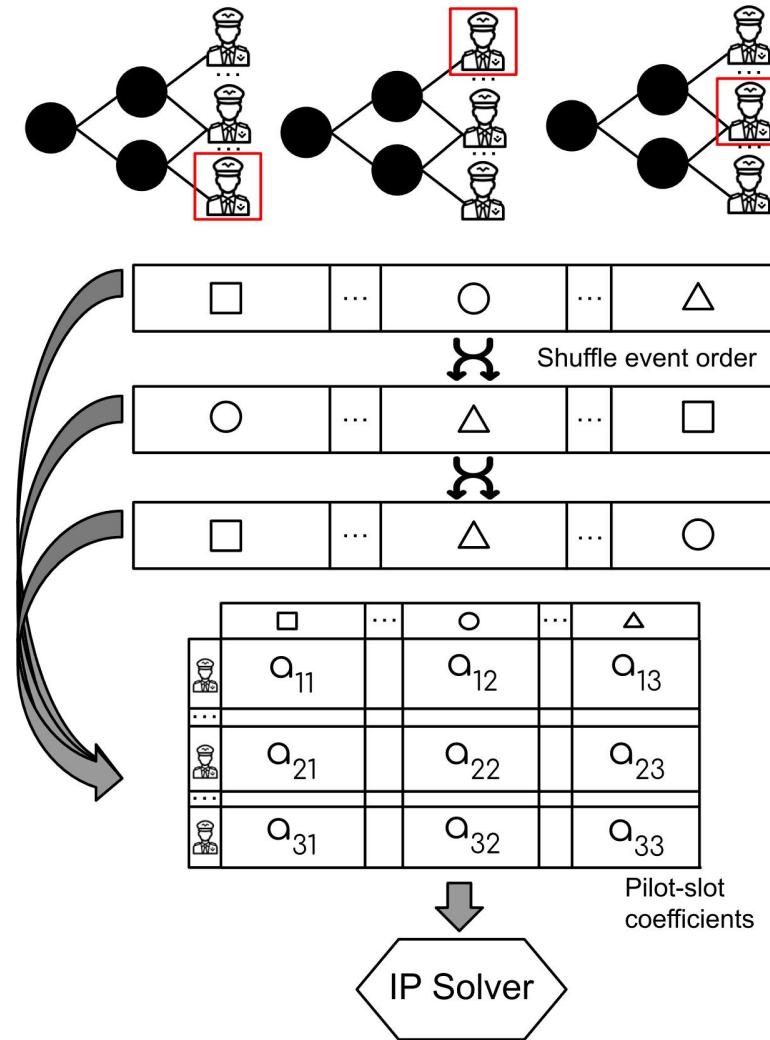
Extracting Probability Weights

- Monte Carlo Approach:
 - Approximate the average weights across all possible orders of scheduling the slots.
 - We randomly shuffle the order of slots that the RL scheduler had to assign and then recorded the probability weights at each step of the scheduling process.
 - Run this process 'n' times to get 'n' total probability weight values for each pilot-slot pair.

Extracting Probability Weights

- Blank Slate Approach
 - Exploit the fact that the probability weights for the pilots produced by the first slot are not dependent on any previous actions taken.
 - Thus, make each slot as the first slot to get weights that are not dependent on previous decisions made.
 - To do so, for each slot 's' in the fixed order, initialize a new RL agent with the same underlying neural network, cutting out all of the states that occur before 's' and then extract the probability weight values for the pilots on that slot.

NICE: Neural Network IP Coefficient Extraction



Reward Structure for Robustness

- Reward of $(b + 1)$ whenever the RL agent places a pilot on an event that forms a buffer of length b with the pilot's most recent event.
 - Add $+1$ term to reward the agent for making a placement, regardless of buffer.
- Reward of $(T - 1)$ when it places a pilot with no previous events scheduled.
- Reward of $+25$ when all slots in an episode are scheduled.
- Reward of -10 when it is unable to schedule all events in an episode.

Experiments

1. Generate 1 week's worth of flights
2. Schedule with baseline IP, RL scheduler, Buffer IP, and NICE
3. One day into the schedule, delay $f\%$ of the flights that had not already left.
 - a. 100 trials each
4. Fix delays with disruption-minimizing IP.
5. Count disruptions with each scheduling method.

Results

% Flights Delayed	Number of Disruptions			
	NICE	Baseline IP	RL	Buffer IP
25	0.34 ± 0.71	0.61 ± 1.07	32.6 ± 7.33	0 ± 0
50	0.67 ± 0.92	1.16 ± 1.55	27.1 ± 7.13	0 ± 0
75	0.66 ± 0.99	1.13 ± 1.73	23.0 ± 6.34	0 ± 0
100	0.63 ± 0.82	1.06 ± 1.50	17.9 ± 6.29	0 ± 0

Results

< 0.85 seconds on average

% Flights Delayed	Number of Disruptions			
	NICE	Baseline IP	RL	Buffer IP
25	0.34 ± 0.71	0.61 ± 1.07	32.6 ± 7.33	0 ± 0
50	0.67 ± 0.92	1.16 ± 1.55	27.1 ± 7.13	0 ± 0
75	0.66 ± 0.99	1.13 ± 1.73	23.0 ± 6.34	0 ± 0
100	0.63 ± 0.82	1.06 ± 1.50	17.9 ± 6.29	0 ± 0

Results

% Flights Delayed	Number of Disruptions		
	NICE	Baseline IP	RL
25	1.99 \pm 1.99	2.99 \pm 2.68	59.2 \pm 13.1
50	3.17 \pm 2.27	5.01 \pm 4.33	51.2 \pm 12.2
75	3.32 \pm 2.33	6.32 \pm 5.64	43.1 \pm 12.4
100	2.81 \pm 2.23	4.70 \pm 4.49	35.4 \pm 10.5

Scheduling density of 2

Results

1.85 to 1.90 seconds on average

% Flights Delayed	Number of Disruptions		
	NICE	Baseline IP	RL
25	1.99 ± 1.99	2.99 ± 2.68	59.2 ± 13.1
50	3.17 ± 2.27	5.01 ± 4.33	51.2 ± 12.2
75	3.32 ± 2.33	6.32 ± 5.64	43.1 ± 12.4
100	2.81 ± 2.23	4.70 ± 4.49	35.4 ± 10.5

Scheduling density of 2

Buffer IP timed out after 90 minutes in each scenario.

The Future

- NICE is highly generalizable. Can be used whenever there is an isomorphism between an IP and RL solution.
 - This is a large class of problems!
 - Want to validate in a variety of domains
 - What types of problems can NICE best solve?
- Use to approximate non-linear constraints and objective functions
- Explore other, potentially more sophisticated, weight extraction methods
- How can it be adapted to increase its performance?

NICE

- Uses reinforcement learning to approximate and simplify computationally difficult problems
 - Leverages knowledge learned by reinforcement learning
 - Utilizes global outlook from integer programming
- Applied to robust flight scheduling