# 4.3. Advanced aspects of RL with Planning

Ángel Aso Mollar
Eva Onaindía de la Rivaherrera

Planificación Inteligente

Curso 2023-2024

Máster Universitario en Inteligencia Artificial,
Reconocimiento de Formas e Imagen Digital

## Contents

## RL approximators

## Deep Reinforcement Learning

- We can approximate any RL element by means of any function approximator: random forest, multivariate regression, decission trees, **neural networks**...
- **Value approximation**: Approximate $v_\pi$ or $q_\pi$. This methods yield deterministic policies.
- **Policy approximation**: Methods that directly approximate the policy $\pi$, using the notion of stochastic policies.

## Policy Gradients: REINFORCE

◇ If we want to define a policy approximator $\pi_\theta$, we can formalize the simplified objective function $J(\theta)$ as the expected reward (remember, $p$ is unknown!):

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} R(\tau) = \sum_\tau p(\tau;\theta) R(\tau)$$

◇ The objective is to maximize the expected reward: $\max_\theta J(\theta)$, so we can compute the gradient of this function and use gradient optimization methods.

$$\nabla_\theta J(\theta) = \sum_\tau \nabla_\theta p(\tau;\theta) R(\tau) = \sum_\tau p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} R(\tau) =$$

$$= \sum_\tau p(\tau;\theta) \nabla_\theta \log p(\tau;\theta) R(\tau) = \mathbb{E}_{\tau \sim \pi_\theta} \nabla_\theta \log p(\tau;\theta) R(\tau)$$

## Policy Gradients: REINFORCE

- But we can define the probability of a trajectory as follows:

$$p(\tau;\theta) = \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t) \cdot \pi_\theta(a_t \mid s_t)$$

- And taking the logarithm, the product becomes a sum. With the gradient, all functions that do not have $\theta$ become zero, so:

$$\nabla_\theta \log p(\tau;\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

- We can simply sample trajectories to estimate the gradient![1]

[1] I'm sorry, I don't have time! Check: `https://towardsdatascience.com/policy-gradients-in-reinforcement-learning-explained-ecec7df94245`

## State of the art algorithms

Vanilla policy gradients algorithm (REINFORCE) has a high variance in its gradient calculation. State-of-the-art algorithms try to solve this problem:

- ◇ **Actor critic methods**: subtract to the REINFORCE gradient a baseline function.
- ◇ **Proximal Policy Optimization**: forces the policy not to highly differ from the current one.

There is no need to only approximate the policy, value functions can also be approximated:

- ◇ **Deep Q-learning**: Approximates the Q function using a loss derived from the Bellman equations.

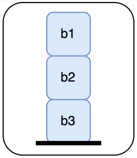
## State embeddings

## INPUT REPRESENTATION

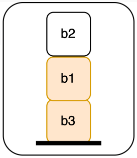
- All function approximators need a numerical representation of the input.
- For the policy/value approximators, we need a *feature vector* that represents all the information of the state.
- In Planning, states are represented in the form of literals, i.e., first-order logic grounded predicates.
- We can use any neuro-symbolic tool in order to produce successful state representations: **Graph Neural Networks** (GNNs), **Neural Logic Machines** (NLMs), etc.
- State representation in Planning is also an open field of study.

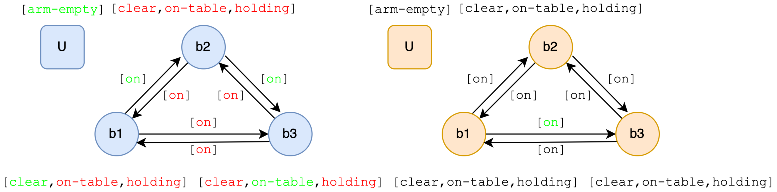
## State-goal graph representation

**Estado actual:**
(arm-empty)
(clear b1)
(on-table b3)
(on b1 b2)
(on b2 b3)



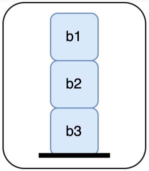
**Estado objetivo:**
(on b1 b3)



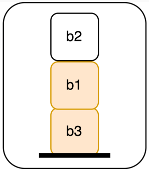

[arm-empty] [clear,on-table,holding]
U
b2
[on]
[on]
[on]
[on]
b1
[on]
b3
[on]
[clear,on-table,holding] [clear,on-table,holding]
[arm-empty] [clear,on-table,holding]
U
b2
[on]
[on]
[on]
[on]
b1
[on]
b3
[on]
[clear,on-table,holding] [clear,on-table,holding]

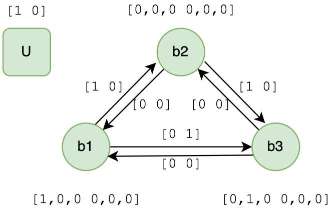
## STATE-GOAL GRAPH REPRESENTATION

**Estado actual:**
```
(arm-empty)
(clear b1)
(on-table b3)
(on b1 b2)
(on b2 b3)
```

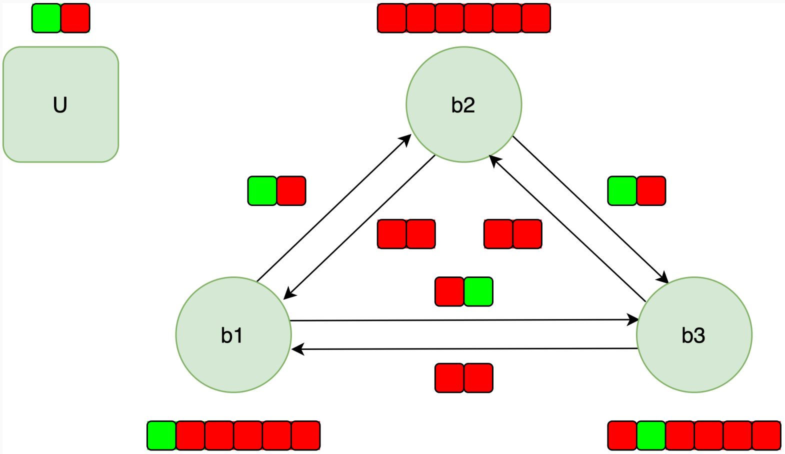


**Estado objetivo:**
```
(on b1 b3)
```

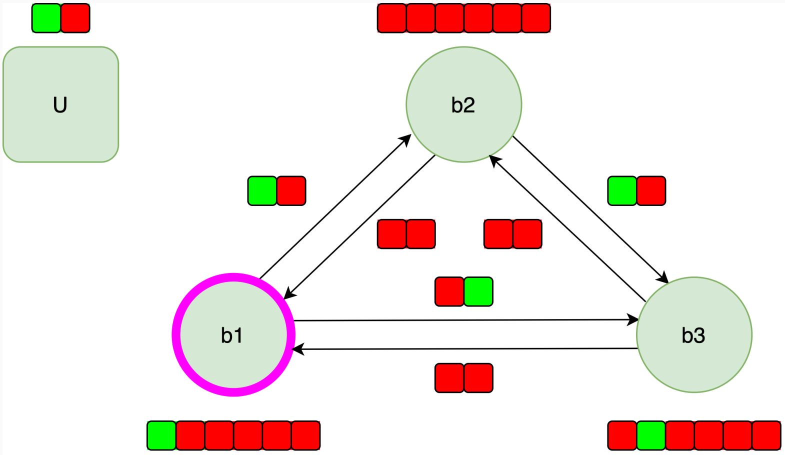




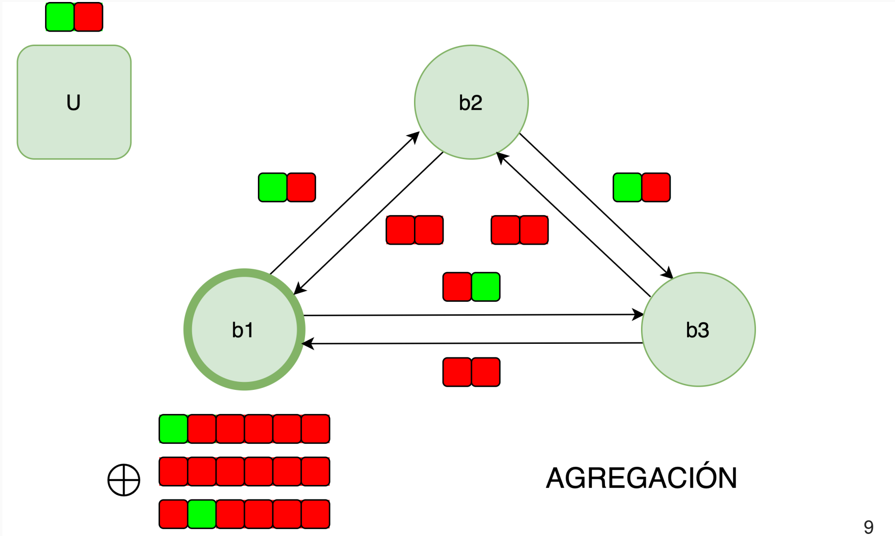
## Graph Neural Networks



## Graph Neural Networks

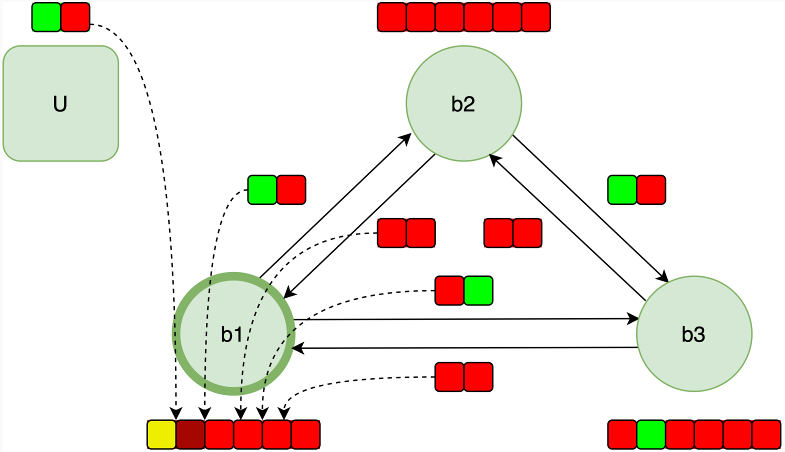


## Graph Neural Networks



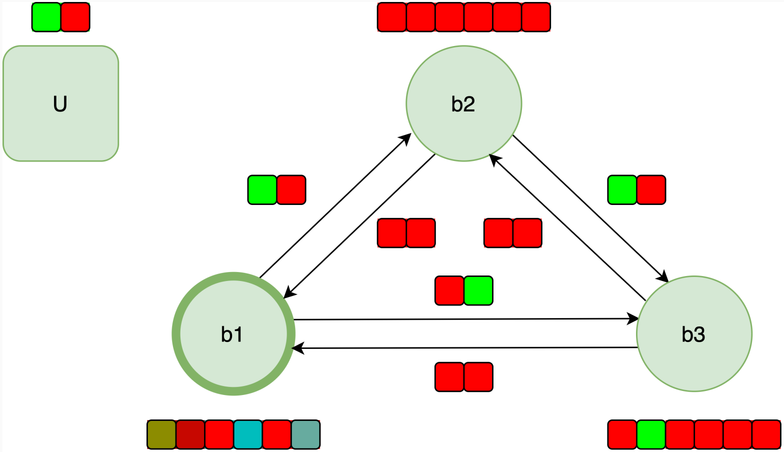
AGREGACIÓN

## Graph Neural Networks

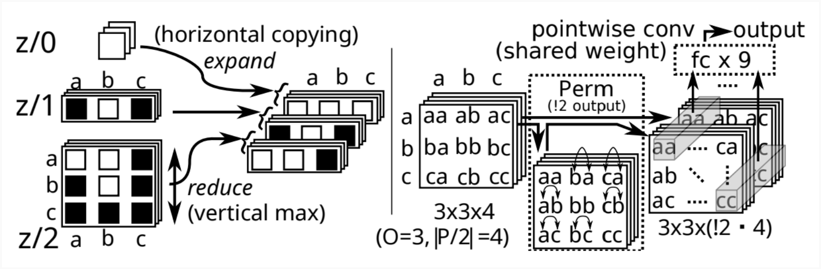


+ INFORMACIÓN GLOBAL Y DE ARISTAS

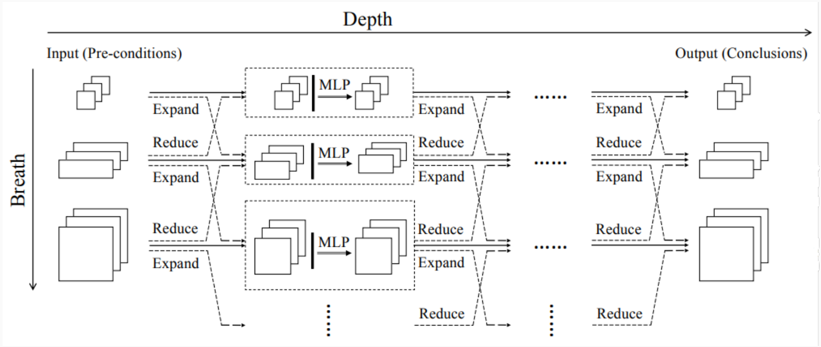
## Graph Neural Networks



## Tensor representation

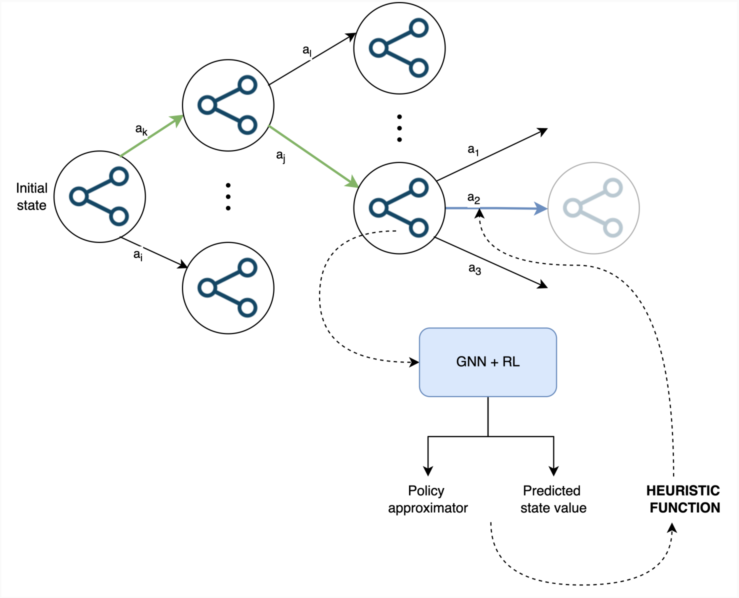


## Neural Logic Machines

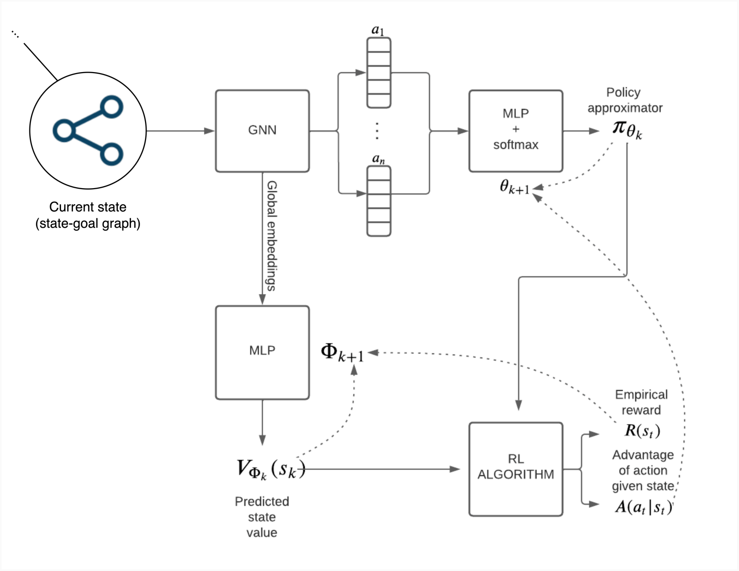


## Advanced integration

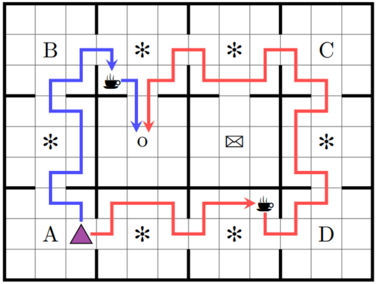
## GENERALIZED PLANNING WITH DEEP RL



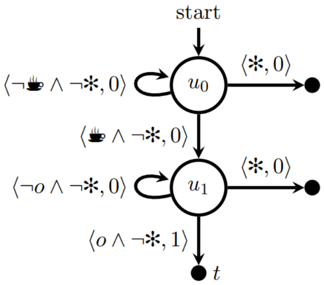
## Generalized planning with Deep RL



## Reward machines



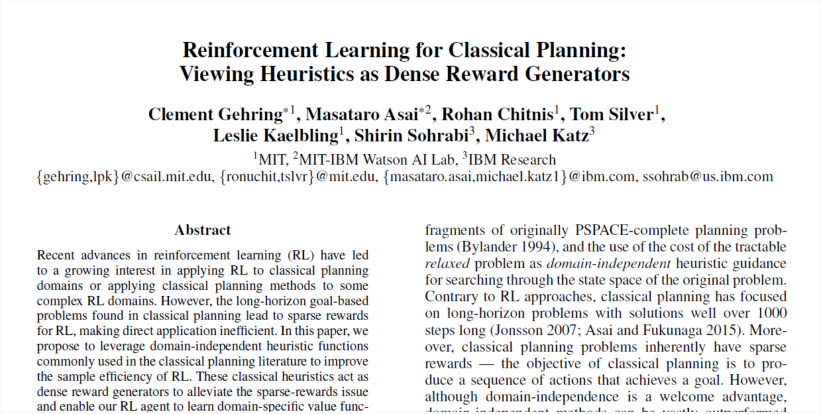
(a) The office gridworld


start
$u_0$
$u_1$
$\langle \neg ☕ \wedge \neg ✻, 0\rangle$
$\langle ✻, 0\rangle$
$\langle ☕ \wedge \neg ✻, 0\rangle$
$\langle \neg o \wedge \neg ✻, 0\rangle$
$\langle ✻, 0\rangle$
$\langle o \wedge \neg ✻, 1\rangle$
$t$

(b) A simple reward machine

https://doi.org/10.48550/arXiv.2010.03950

## VIEWING HEURISTICS AS DENSE REWARD GENERATORS



**Reinforcement Learning for Classical Planning: Viewing Heuristics as Dense Reward Generators**

**Clement Gehring*[1], Masataro Asai*[2], Rohan Chitnis[1], Tom Silver[1], Leslie Kaelbling[1], Shirin Sohrabi[3], Michael Katz[3]**

[1]MIT, [2]MIT-IBM Watson AI Lab, [3]IBM Research

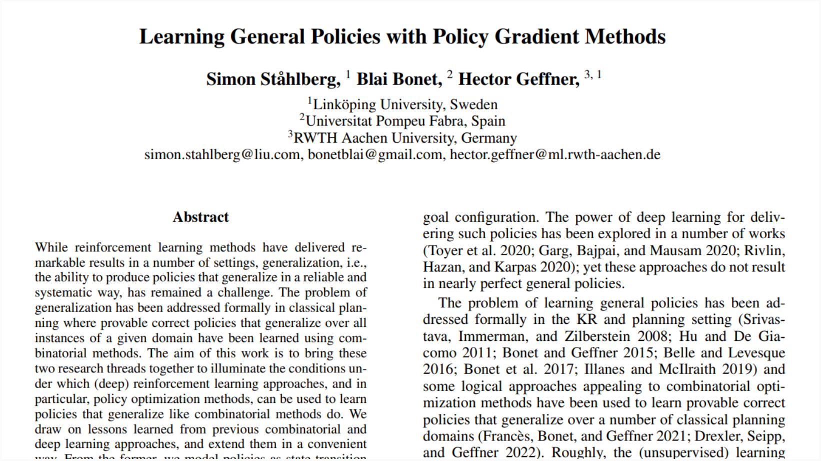
{gehring,lpk}@csail.mit.edu, {ronuchit,tslvr}@mit.edu, {masataro.asai,michael.katz1}@ibm.com, ssohrab@us.ibm.com

**Abstract**

Recent advances in reinforcement learning (RL) have led to a growing interest in applying RL to classical planning domains or applying classical planning methods to some complex RL domains. However, the long-horizon goal-based problems found in classical planning lead to sparse rewards for RL, making direct application inefficient. In this paper, we propose to leverage domain-independent heuristic functions commonly used in the classical planning literature to improve the sample efficiency of RL. These classical heuristics act as dense reward generators to alleviate the sparse-rewards issue and enable our RL agent to learn domain-specific value func-

fragments of originally PSPACE-complete planning problems (Bylander 1994), and the use of the cost of the tractable *relaxed* problem as *domain-independent* heuristic guidance for searching through the state space of the original problem. Contrary to RL approaches, classical planning has focused on long-horizon problems with solutions well over 1000 steps long (Jonsson 2007; Asai and Fukunaga 2015). Moreover, classical planning problems inherently have sparse rewards — the objective of classical planning is to produce a sequence of actions that achieves a goal. However, although domain-independence is a welcome advantage,

## LEARNING GENERAL POLICIES



**Learning General Policies with Policy Gradient Methods**

**Simon Ståhlberg,** [1] **Blai Bonet,** [2] **Hector Geffner,** [3, 1]
[1]Linköping University, Sweden
[2]Universitat Pompeu Fabra, Spain
[3]RWTH Aachen University, Germany
simon.stahlberg@liu.com, bonetblai@gmail.com, hector.geffner@ml.rwth-aachen.de

**Abstract**

While reinforcement learning methods have delivered remarkable results in a number of settings, generalization, i.e., the ability to produce policies that generalize in a reliable and systematic way, has remained a challenge. The problem of generalization has been addressed formally in classical planning where provable correct policies that generalize over all instances of a given domain have been learned using combinatorial methods. The aim of this work is to bring these two research threads together to illuminate the conditions under which (deep) reinforcement learning approaches, and in particular, policy optimization methods, can be used to learn policies that generalize like combinatorial methods do. We draw on lessons learned from previous combinatorial and deep learning approaches, and extend them in a convenient

goal configuration. The power of deep learning for delivering such policies has been explored in a number of works (Toyer et al. 2020; Garg, Bajpai, and Mausam 2020; Rivlin, Hazan, and Karpas 2020); yet these approaches do not result in nearly perfect general policies.

The problem of learning general policies has been addressed formally in the KR and planning setting (Srivastava, Immerman, and Zilberstein 2008; Hu and De Giacomo 2011; Bonet and Geffner 2015; Belle and Levesque 2016; Bonet et al. 2017; Illanes and McIlraith 2019) and some logical approaches appealing to combinatorial optimization methods have been used to learn provable correct policies that generalize over a number of classical planning domains (Francès, Bonet, and Geffner 2021; Drexler, Seipp, and Geffner 2022). Roughly, the (unsupervised) learning

## Open challenges

## OPEN CHALLENGES

- **Parallel planning**: introducing parallel action application in RL (maybe to generalized planning).
- **New state or action embedding generation**: straightforward structural embeddings.
- **Model learning**: (PDDL?) model learning from traces.