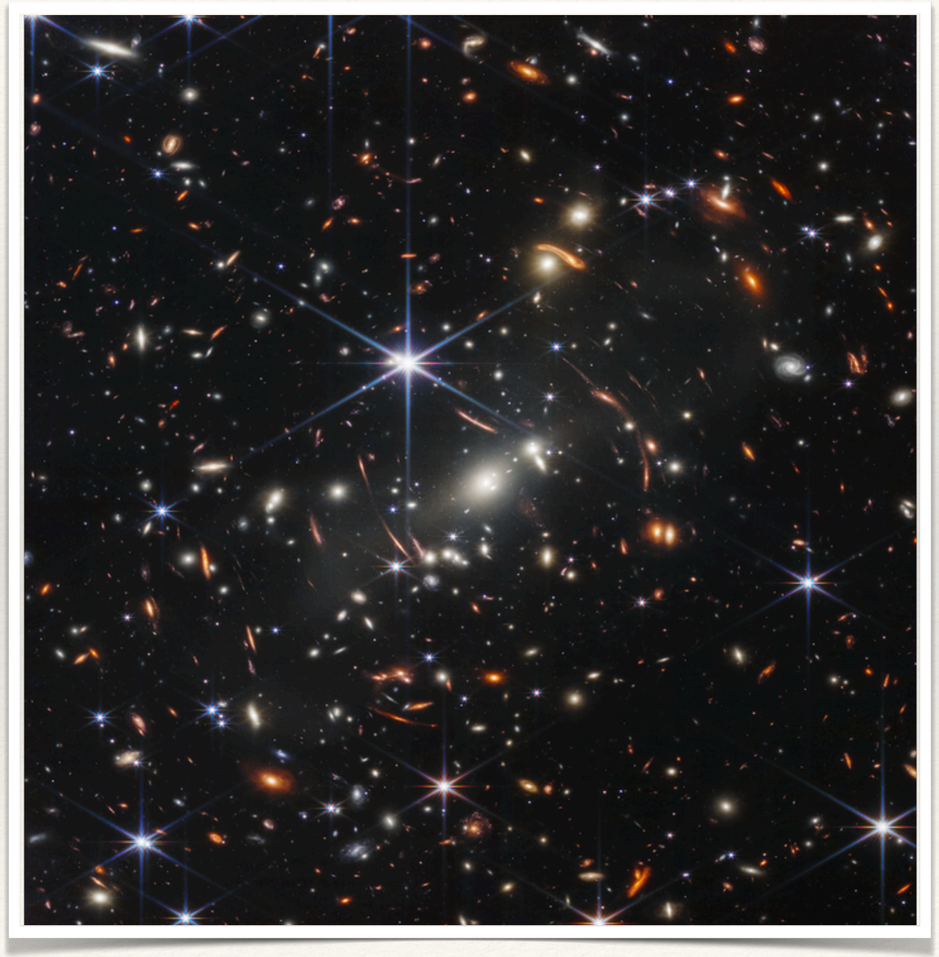
Progress Presentation

April 3rd - May 12th

Agenda

New algorithm for solving I-POMDPs with continuous observation spaces to approximate rationality of civilisations

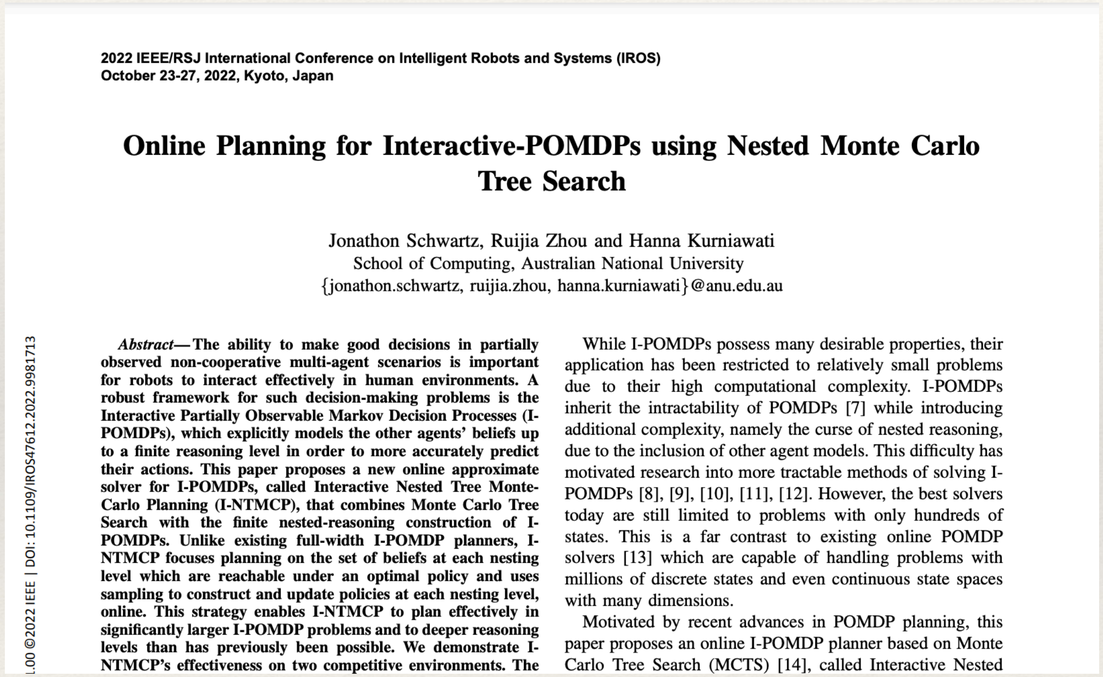
What kind of strategies do civilisations in the universe employ to ensure their survival, and how do these strategies change over time and space?



JWST's deep field of SMACS 0723

Origins of the new algorithm

"I-NTMCP"



2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Online Planning for Interactive-POMDPs using Nested Monte Carlo Tree Search

Jonathon Schwartz, Ruijia Zhou and Hanna Kurniawati
School of Computing, Australian National University
{jonathon.schwartz, ruijia.zhou, hanna.kurniawati}@anu.edu.au

978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981713

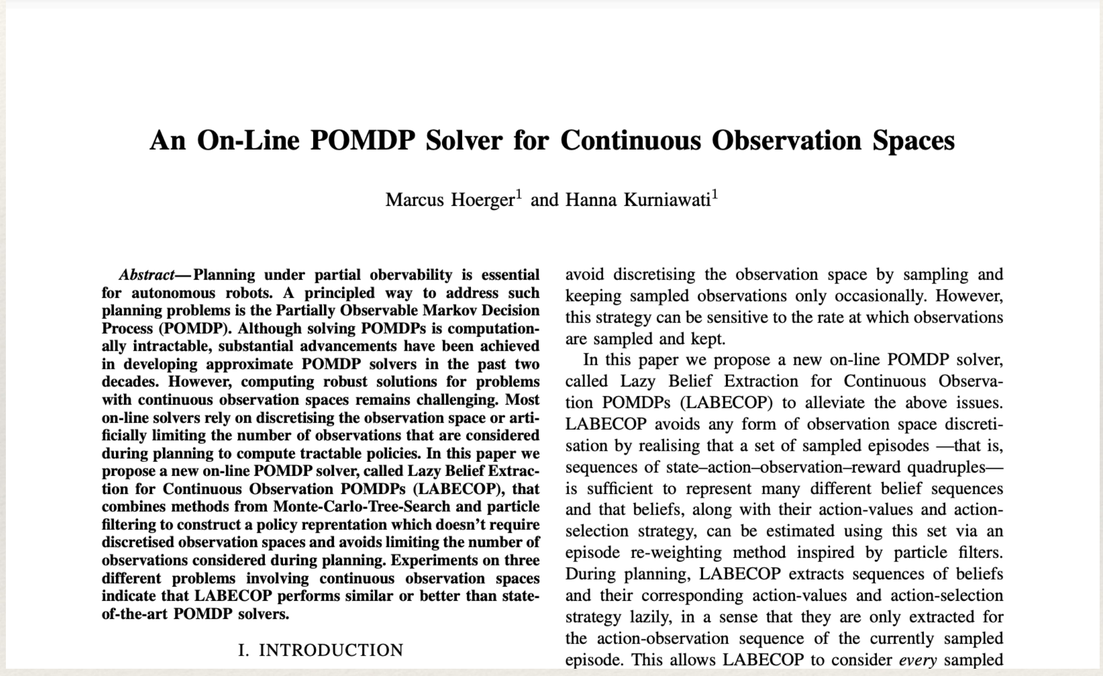
***Abstract*— The ability to make good decisions in partially observed non-cooperative multi-agent scenarios is important for robots to interact effectively in human environments. A robust framework for such decision-making problems is the Interactive Partially Observable Markov Decision Processes (I-POMDPs), which explicitly models the other agents' beliefs up to a finite reasoning level in order to more accurately predict their actions. This paper proposes a new online approximate solver for I-POMDPs, called Interactive Nested Tree Monte-Carlo Planning (I-NTMCP), that combines Monte Carlo Tree Search with the finite nested-reasoning construction of I-POMDPs. Unlike existing full-width I-POMDP planners, I-NTMCP focuses planning on the set of beliefs at each nesting level which are reachable under an optimal policy and uses sampling to construct and update policies at each nesting level, online. This strategy enables I-NTMCP to plan effectively in significantly larger I-POMDP problems and to deeper reasoning levels than has previously been possible. We demonstrate I-NTMCP's effectiveness on two competitive environments. The**

While I-POMDPs possess many desirable properties, their application has been restricted to relatively small problems due to their high computational complexity. I-POMDPs inherit the intractability of POMDPs [7] while introducing additional complexity, namely the curse of nested reasoning, due to the inclusion of other agent models. This difficulty has motivated research into more tractable methods of solving I-POMDPs [8], [9], [10], [11], [12]. However, the best solvers today are still limited to problems with only hundreds of states. This is a far contrast to existing online POMDP solvers [13] which are capable of handling problems with millions of discrete states and even continuous state spaces with many dimensions.

Motivated by recent advances in POMDP planning, this paper proposes an online I-POMDP planner based on Monte Carlo Tree Search (MCTS) [14], called Interactive Nested

https://rdl.cecs.anu.edu.au/papers/iros22_ntmcp.pdf

"LABECOP"



An On-Line POMDP Solver for Continuous Observation Spaces

Marcus Hoerger[1] and Hanna Kurniawati[1]

***Abstract*— Planning under partial obervability is essential for autonomous robots. A principled way to address such planning problems is the Partially Observable Markov Decision Process (POMDP). Although solving POMDPs is computationally intractable, substantial advancements have been achieved in developing approximate POMDP solvers in the past two decades. However, computing robust solutions for problems with continuous observation spaces remains challenging. Most on-line solvers rely on discretising the observation space or artificially limiting the number of observations that are considered during planning to compute tractable policies. In this paper we propose a new on-line POMDP solver, called Lazy Belief Extraction for Continuous Observation POMDPs (LABECOP), that combines methods from Monte-Carlo-Tree-Search and particle filtering to construct a policy reprentation which doesn't require discretised observation spaces and avoids limiting the number of observations considered during planning. Experiments on three different problems involving continuous observation spaces indicate that LABECOP performs similar or better than state-of-the-art POMDP solvers.**

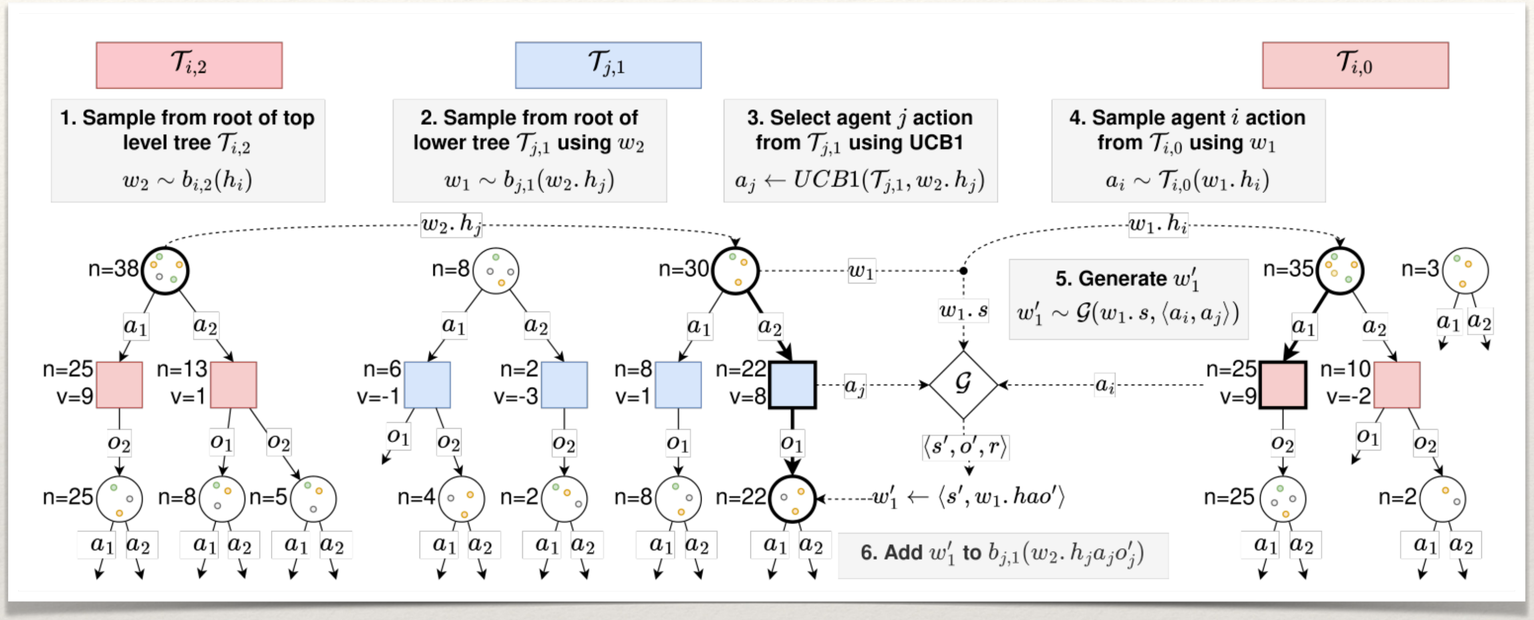
I. INTRODUCTION

avoid discretising the observation space by sampling and keeping sampled observations only occasionally. However, this strategy can be sensitive to the rate at which observations are sampled and kept.

In this paper we propose a new on-line POMDP solver, called Lazy Belief Extraction for Continuous Observation POMDPs (LABECOP) to alleviate the above issues. LABECOP avoids any form of observation space discretisation by realising that a set of sampled episodes —that is, sequences of state–action–observation–reward quadruples— is sufficient to represent many different belief sequences and that beliefs, along with their action-values and action-selection strategy, can be estimated using this set via an episode re-weighting method inspired by particle filters. During planning, LABECOP extracts sequences of beliefs and their corresponding action-values and action-selection strategy lazily, in a sense that they are only extracted for the action-observation sequence of the currently sampled episode. This allows LABECOP to consider *every* sampled

https://rdl.cecs.anu.edu.au/papers/icra21_ctsObs.pdf

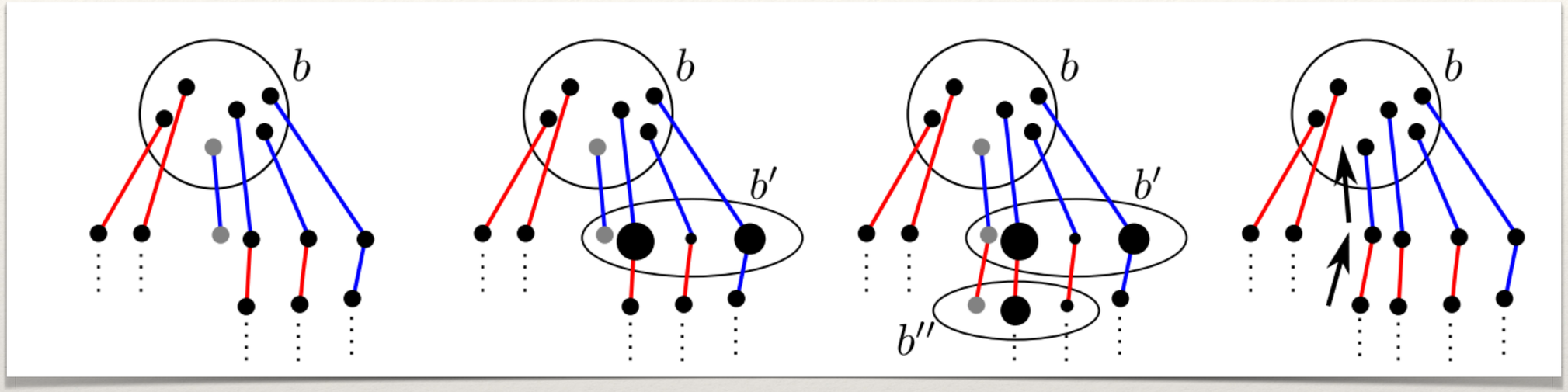
I-NTMCP



- two agents, *i* and *j*
- we are planning for agent *i*, who is at level 2

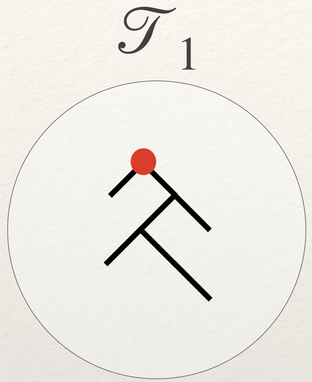
- each particle is a "history state" $w^t = (s^t, h^t)$ where s^t is a state at time t and $h^t = (h_i^t, h_j^t)$ is a joint history where $h_k^t = (a_k^1, o_k^2, a_k^2, o_k^3, \ldots, a_k^{t-1}, o_k^t)$ is an agent history
- tree nodes correspond to agent histories

LABECOP



- two actions, red and blue
- starts at the current belief *b* (can be represented in any way)
    1. sample a state from the belief
    2. choose* an action and propagate state
    3. weight particles corresponding to that action history by weights of source particles and observation probabilities to create a new belief
    4. repeat

New algorithm – overview

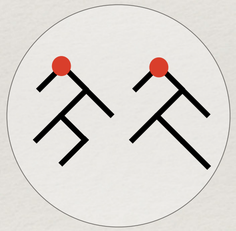


$\mathcal{T}_1$

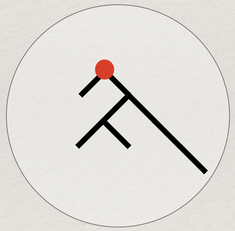
the same hierarchical tree structure as I-NTMCP

$\Rightarrow \sum_{d=0}^{L} (n-1)^d = \mathcal{O}(n^L)$ trees for n agents at max. level L

Level 2



$\mathcal{T}_{1,2}$



$\mathcal{T}_{1,3}$

Level 1

Represents 1's beliefs
about 3's beliefs
about 2's beliefs

$\mathcal{T}_{1,2,1}$

$\mathcal{T}_{1,2,3}$

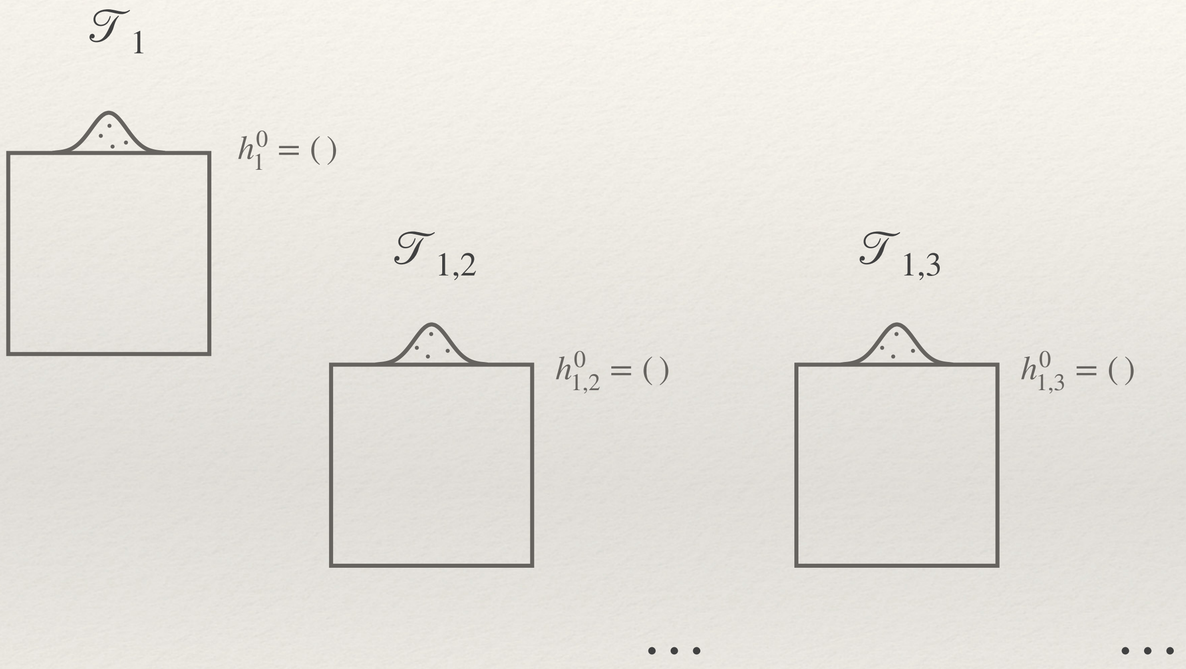
$\mathcal{T}_{1,3,1}$

$\mathcal{T}_{1,3,2}$

Level 0

● = root node

New algorithm - initialisation

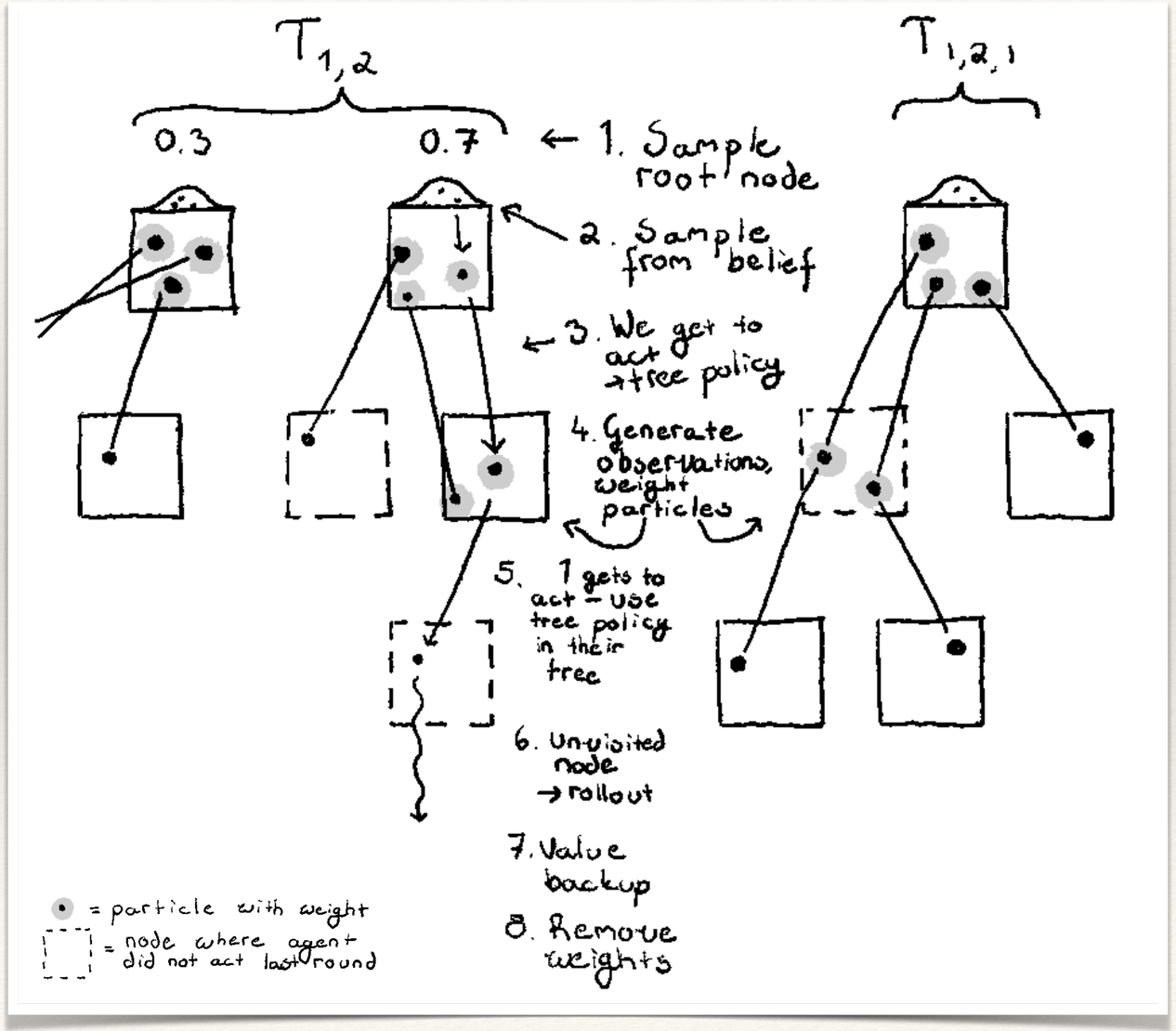


- Each node in a tree corresponds to an agent action history
- A node contains a set of particles
  - particles are **history states**: they contain an environment state and a joint action history
- In addition, root nodes contain a separate set of "belief particles" which are history states as well, but have weights
- Initially all trees have a single root node (empty agent action history), but in general the lower-level trees can have multiple root nodes
  - top level tree (here $\mathcal{T}_1$) always has a unique root node

New algorithm - planning

- planning consists of expanding each tree a given numbers of times

- bottom up: first we plan at level 0, then level 1, and so on



New algorithm - planning

- The tree policy resembles MCTS's UCT method, but has to be adapted slightly
  - Actions are always chosen from beliefs, but since the observation space is continuous, in general we will never visit the same belief twice
- When the particles p in a node N all have weights $b(p)$, the particles represent a belief b. Calculate:

  $N_+(b) =$ number of particles with weight > 0 under b

  $W(b,a) = \sum_{p \in N, p.a = a} b(p) =$ total weight of particles that were next propagated with a

  $\tilde{W}(b,a) = W(b,a) N_+(b)$

  $Q(b,a) = \frac{1}{W(b,a)} \sum_{p \in N : p.a = a} b(p) V(p) =$ weighted average of particle values

- Then the chosen action is one that maximises $Q(b,a) + c\sqrt{\frac{\log N_+(b)}{\tilde{W}(b,a)}}$ (c is exploration constant)

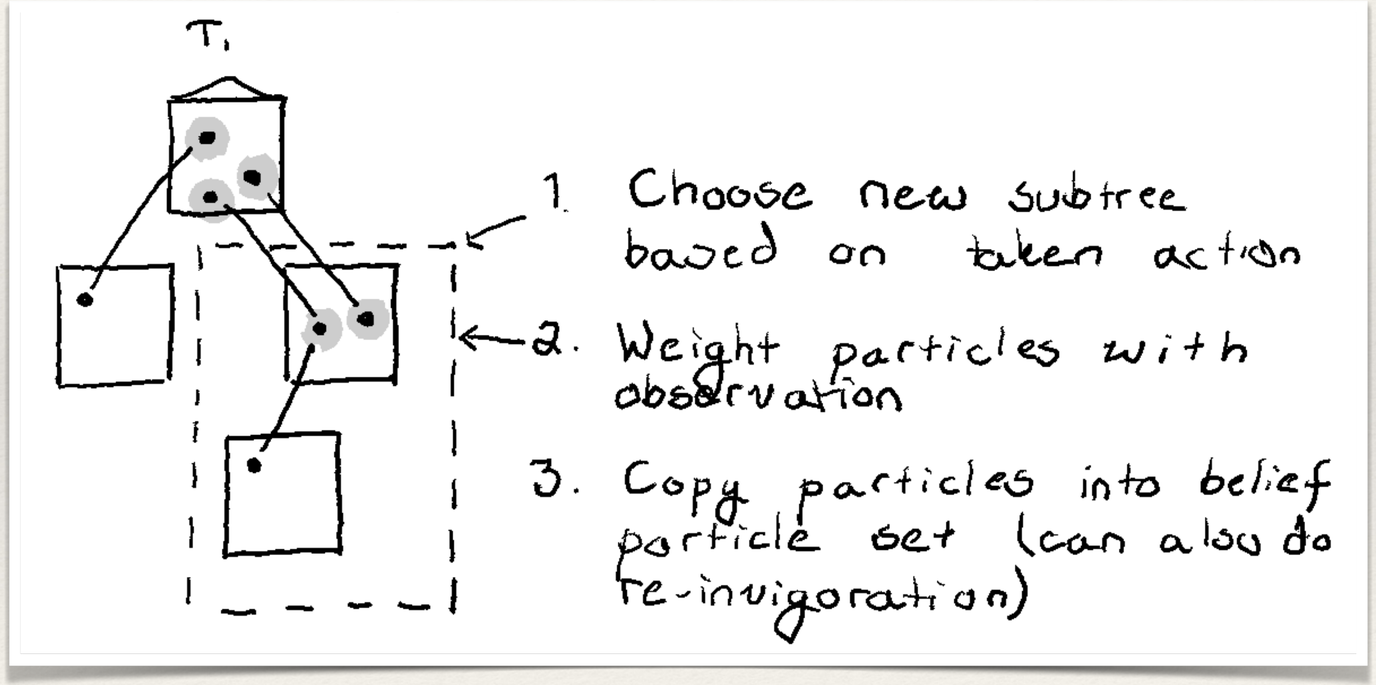
New algorithm - belief update

- Agent takes an action and receives an observation
- Belief update happens top down: first the top-level tree, then the trees below them, etc.
- Need to:

i. find weights for new root nodes

ii. create beliefs at each new root node

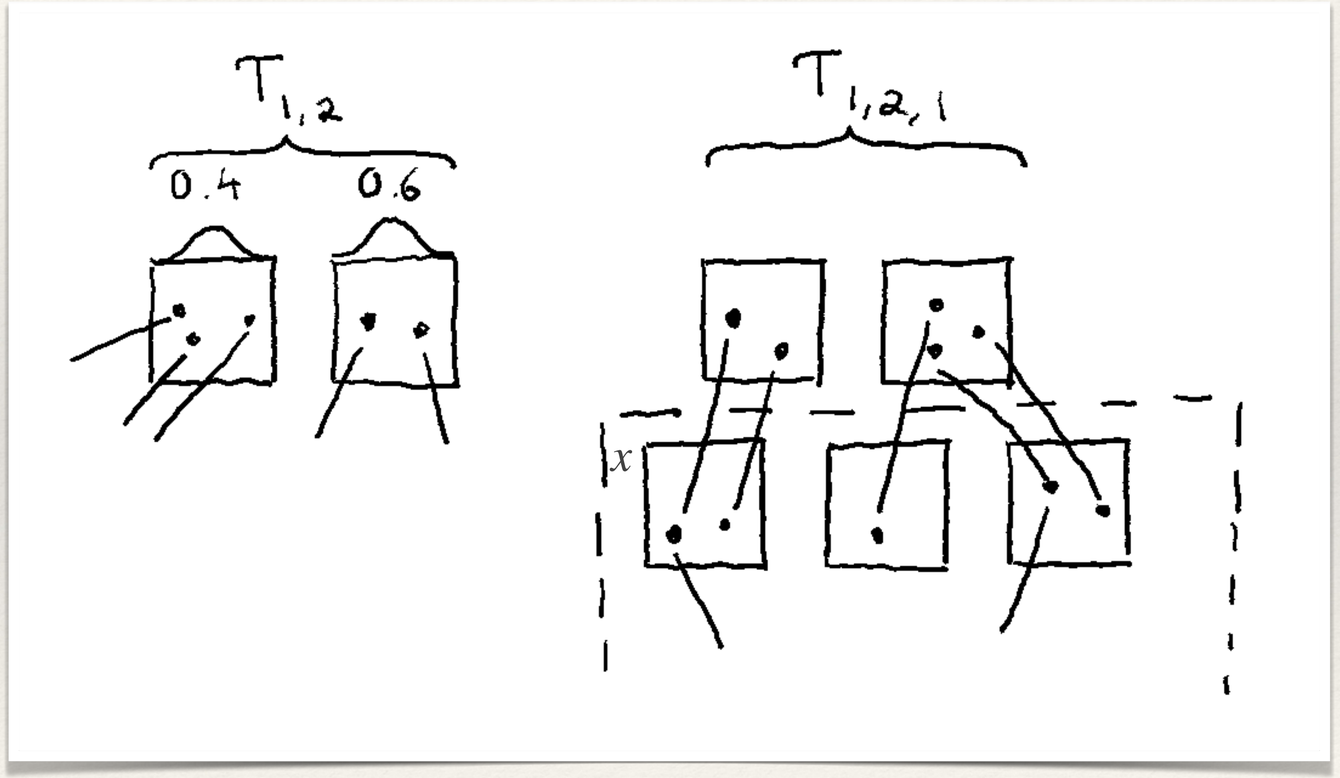
Belief update for the top-level tree



New algorithm - belief update (lower-level trees)

i. to find a weight for a new root node *x*:

- 1. calculate proportion of particles in each parent tree root node where joint history matches agent history of *x*
- 2. weight proportions by parent tree root node weights and sum

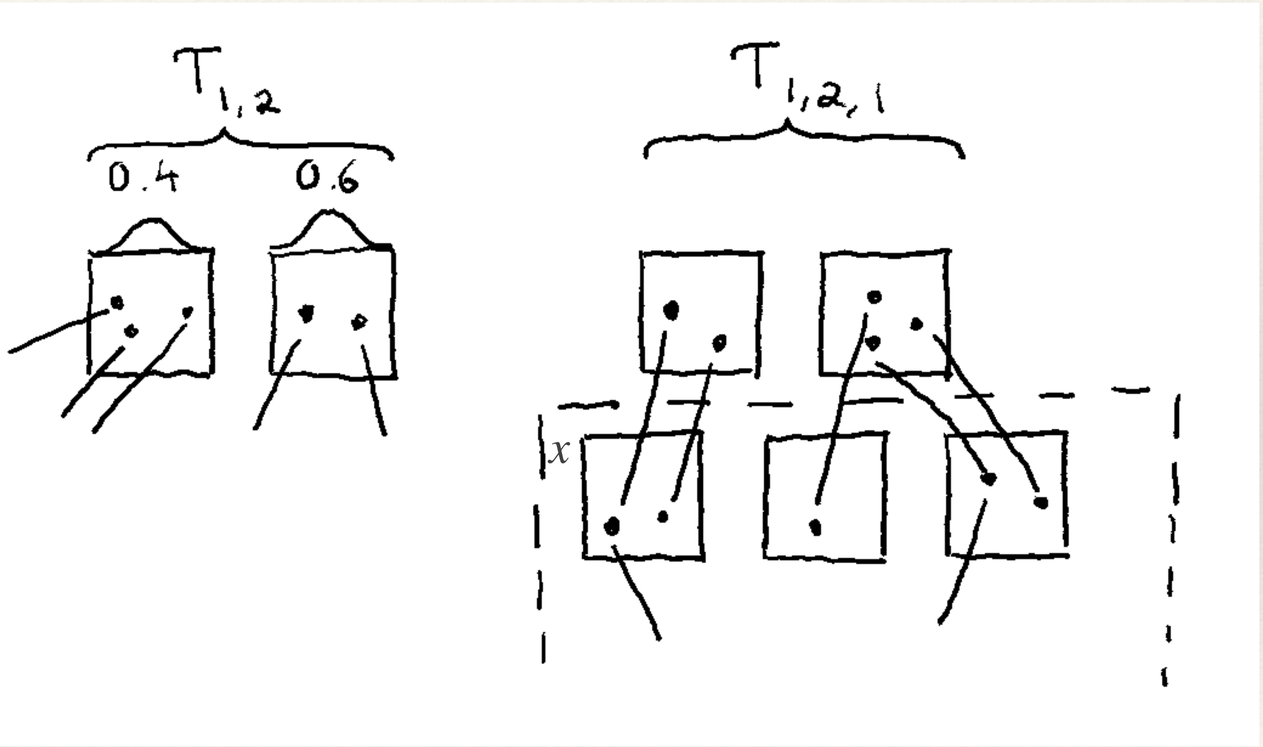


New algorithm - belief update (lower-level trees)

To create a belief for new root node x, repeat n times:

- 1. sample parent tree root node (with weights)
- 2. from the chosen node, sample (with weights) a particle from among the particles matching x
- 3. generate observation and weight particles in x
- 4. sample (with weights) a particle from x and increase its count by 1

- finally, divide counts by n to get approximate weights for particles
- root belief particle set can then be created as before



Results

- Soon™
- Algorithm is implemented, but still has some bugs
- 90% confident I can send the first results this weekend

Some other updates

- Observations now include the agent's own state (since they should always know their own state)
- Code now supports starting the civilisations at different levels of development
- I followed a supercomputing tutorial at Surf
  - will apply for time on Snellius once I know how much I need
- Add me back to thesis group on Slack, free trial ran out
- Continuing after thesis?