Algorithm 2: DRTDP for agent i

DRTDP(i)
  $\forall s \in S, a \in A, Q_i(s,a) \leftarrow 0, V_i(s) \leftarrow 0$
  while $true$ **do**
    process-messages()
    if $s_c^i \neq null$ **then**
      advance-trajectory()
process-message()
  foreach *Message* $m = \langle s, t, v, j \rangle$ **do**
    if $m.type =$ *Q-value request* **then**
      send to $m.j$ $\langle m.s, Q$-value response, $V_i(s), i \rangle$
    if $m.type =$ *Q-value response* **then**
      if $m.v < V_i(s)$ **then**
        $V_i(s) = m.v, best_i(s) \leftarrow m.j$
    if $m.type =$ *trajectory* **then**
      $s_c^i \leftarrow m.s$
advance-trajectory()
  $a^* = argmin_a Q_i(s_c^i, a)$
  update(s_c^i, a^*)
  sample s' from $tr(s_c^i, a^*, \cdot)$
  $s_c^i \leftarrow$**choose-next-agent**(s')
choose-next-agent(s)
  if $s \subseteq G$ **then**
    broadcast & wait $\langle s_0, Q$-value request, $\times, i \rangle$
    $s \leftarrow s_0$
  if $best(s) = i$ **then return** s
  else send to $best(s)$ $\langle s,$ trajectory, $\times, i \rangle$, **return** $null$
update(s, a)
  foreach $s' \in tr(s, a, \cdot)$ **do**
    broadcast & wait $\langle s', Q$-value request, $\times, i \rangle$
  $Q_i(s,a) = C(s,a) + \sum_{s'} tr(s,a,s') V_i(s')$
  $V_i(s) \leftarrow min_{a'} Q_i(s, a')$

contain the current values for the states in the message.

In the complete DRTDP, at each step, a single agent is responsible for advancing the trajectory. This agent requests all other agents to send it their estimated cost to the goal from the current state for their best action. The agent sending the best expected cost is chosen to execute its action and select the next state. This is repeated until the goal is reached. Each agent i maintains the functions $Q_i(s,a)$ and $V_i(s)$.

Algorithm **??** presents a simplified version of our distributed RTDP method. Each agent continuously processes all received messages (line **??**). A message m is a tuple $\langle s, t, v, i \rangle$ where s is a state, represented by the public facts and the private state indexes of all agents, v is a (possibly empty) value, i the sending agent, and t is the message type. There can be 3 types of messages: (1) Q-value request: agent i is requesting all agents to send their values for the state s. (2) Q-value response: agent i is answering a Q-value request message with its value for the state s. The receiving agent checks if it has received a better value, and if so, changes $V_i(s)$, also recording the sending agent in $best_i(s)$ (line **??**). (3) Trajectory: agent i transfers responsibility of advancing the trajectory from s to the receiving agent.

The private variable s_c^i maintains the current state of the trajectory for agent i, which can be $null$, if the trajectory is currently under the responsibility of another agent. If

Algorithm 3: PS-RTDP for agent i

advance-trajectory()
  $a^* = \min_a Q_i(s_c^i, a)$
  sample s' from $tr(s_c^i, a^*, \cdot)$
  if *private cycle detected* **then**
    restart trajectory
  if a^* *is a private action* **then**
    local-update(s_c^i, a^*), $s_c^i \leftarrow s'$
  else
    update(s_c^i, a^*), $s_c^i \leftarrow$**choose-next-agent**(s')
local-update(s, a)
  $Q_i(s,a) = C(s,a) + \sum_{s'} tr(s,a,s') \min_{a'} Q_i(s', a')$
  $V_i(s) \leftarrow \min_a Q_i(s,a)$

$s_c^i \neq null$, then i is responsible for advancing the trajectory (line **??**). When receiving a trajectory message, s_c^i is set to the received state. When agent i receives responsibility for a trajectory, it first selects its best action a^* (line **??**). At this point, we can limit our attention to the actions of agent i, because we already compared the values from all other agents, and i reported the best expected cost.

We then update the Q-function and the value function. The only Q-value that changes is $Q_i(s_c^i, a^*)$ of agent i whose action is executed. Hence, agent i requests (line **??**) all other agents to send their values for all possible next states after executing a^* at s_c^i. After all values have been received, we update $Q_i(s_c^i, a^*)$ and $V_i(s_c^i)$ (line **??**).

After updating the value function, the agent selects the next possible state s', and then must choose the agent that receives the responsibility to advance the trajectory for s'. First, if s' is a goal state, then the agent must start a new trajectory. To do that, the agent sends a request to all agents for their best value for the initial state s_0. If s' is not a goal state, recall that during the value function update the agent has already requested values from all other agents for all possible next states, including s' (line **??**). Hence, agent i already maintains the best agent to handle s' in $best_i(s')$. If the best next agent is i, then it sets s_c^i to s'. Otherwise, it sends a trajectory message to $best_i(s')$, transferring responsibility for the trajectory, and sets s_c^i to $null$. This algorithm results in identical trajectories to the ones generated by RTDP on the joint problem. Thus, the cost added for preserving privacy is only the cost of the message passing mechanism.

Public Synchronization RTDP The approximate version of DRTDP, which we call Public Sync RTDP (PS-RTDP), relies on the intuition that often there is no need to interleave the private actions of different agents. That is, as in MAFS, following a private action of agent i, the next action should also be of agent i. Hence, in PS-RTDP, an agent chosen to execute the next action continues to execute additional actions until it executes a public action. Then, the agents vote, as in the complete DRTDP on which agent takes ownership of the trajectory and progresses it farther.

Algorithm **??** shows the differences between the algorithms. In PS-RTDP, the agent responsible for advancing a trajectory considers only its own actions. If the selected action is private, then the agent updates the value using only its

Domain	# Actions	# Facts	Best Cost	Expansions × 10^4	Messages × 10^4	# trajectories + restarts	Total Time (sec)
blocks-2-2	26	26	(4.58 / 4.52)	(0.025 / 0.045)	(0.135 / 0.038)	(40 / 40 + 41)	(0.068 / 0.062)
blocks-3-3	129	63	(7.48 / 7.44)	(0.120 / 0.202)	(1.274 / 0.455)	(60 / 90 + 127)	(0.516 / 0.473)
blocks-4-3	315	99	(9.7 / 9.72)	(0.713 / 0.840)	(7.636 / 2.394)	(80 / 130 + 224)	(4.1 / 3.5)
blocks-5-2	422	107	(11.98 / 12.36)	(5.247 / 5.412)	(30.434 / 9.366)	(150 / 380 + 691)	(35.3 / 29.8)
blocks-6-2	746	146	(14.8 / 16.5)	(68.322 / 68.977)	(399.718 / 117.096)	(900 / 2140 + 7057)	(497.8 / 427.7)
depot-2-3	69	49	(11.2 / 11.4)	(0.276 / 0.973)	(2.844 / 3.219)	(90 / 650 + 62)	(1.2 / 3.0)
depot-3-3	149	74	(13.48 / 13.22)	(2.894 / 3.152)	(29.708 / 8.126)	(220 / 160 + 483)	(22.7 / 15.6)
depot-3-4	334	91	(16.76 / 16.76)	(9.273 / 11.765)	(141.352 / 44.033)	(410 / 240 + 2515)	(79.8 / 55.8)
depot-2-5	245	71	(11.4 / 11.14)	(4.342 / 5.050)	(89.343 / 23.417)	(350 / 170 + 1389)	(55.7 / 23.9)
depot-3-5	407	113	(16.78 / 16.78)	(55.045 / 59.236)	(1136.297 / 289.422)	(2330 / 740 + 10437)	(715.4 / 344.8)
logistics-1-3	49	27	(12.78 / 12.62)	(0.371 / 0.191)	(4.208 / 0.164)	(110 / 40 + 102)	(1.1 / 0.217)
logistics-2-3	69	34	(20.38 / 20.78)	(4.203 / 2.516)	(47.815 / 3.494)	(440 / 140 + 792)	(13.9 / 3.5)
logistics-2-4	80	40	(20.82 / 20.84)	(13.437 / 7.174)	(224.140 / 14.711)	(1130 / 170 + 2440)	(54.4 / 10.9)
logistics-2-5	203	70	(27.0 / 21.58)	(122.934 / 40.523)	(2705.937 / 103.567)	(2134 / 220 + 14162)	(848.5 / 81.7)
logistics-3-4	104	48	(26.8 / 27.22)	(152.786 / 109.649)	(2570.205 / 250.775)	(8010 / 770 + 27091)	(720.6 / 186.0)

Table 1: Empirical comparison of the DRTDP and PS-RTDP. Cell format is DRTDP / PS-RTDP.

own Q-values (line **??**), and remains responsible for the trajectory. If a^* is a public action, however, the agent uses the value function update mechanism, and selects the next agent to take responsibility as in Algorithm **??** (line **??**). As all messages are sent by the global update mechanism and the next agent selection process, reducing calls to these phases reduces the number of messages significantly.

PS-RTDP is no longer guaranteed to converge. One reason is that an agent chosen to advance the trajectory may get stuck in a loop of private actions, without being able to reach the goal, or execute any public action. For example, in the Logistics domain, agent 1 may unload a package at location A. As all agents currently have Q-values of 0, the tie breaking mechanism decided that an agent with no access to A takes the trajectory. That agent has no public actions available, and can hence only advance the trajectory by private driving actions, never releasing responsibility to other agents. We therefore add a private cycle detection mechanism (line **??**). If a cycle is detected, we restart the trajectory from s_0. The sensitivity of the cycle detection is important, as cycles may arise due to stochastic effects. Detecting a cycle too soon may make the algorithm ignorant to a possible path towards the goal. Detecting a cycle too late may cause the algorithm to be much slower than the complete DRTDP.

4 Empirical Analysis

We now provide empirical analysis on domains adapted from the CPPP literature, comparing the complete DRTDP and the approximate DRTDP algorithms, implemented in Python. All experiments were run on Google cloud running a Xeon CPU with two 2GHz cores, and 1.8GB user RAM. The domains were adapted by adding stochastic effects. For example, in the blocks world domain, when picking a block it may either be successfully picked, stay where it was, or fall on the table. We also added different capabilities and success probabilities for different agents for completing different tasks, requiring smarter policies for optimization. For example, we added block types, where each arm can lift only a subset of block types, with varying success probabilities.

On each problem we stop the algorithms after 10 RTDP trajectories, and estimate the average cost over 50 policy executions. We terminate once the policy stops improving.

Table **??** shows the results. In many domains the approximate DRTDP generated orders of magnitude fewer messages than the complete version, while converging to policies of similar quality. In all domains both methods expanded a similar number of states, but often the approximate version requires fewer goal reaching trajectories, not counting restarts after loops, that can be much shorter.

In the blocks domain, the approximate RTDP generated about one third of the messages. On the other hand, it often required many more trajectories to converge, especially for the larger problems, and hence resulted in only slightly lower runtime. Only in the largest blocks problem, PS-RTDP resulted in significantly higher average cost. This is because in blocks world there is a relatively small number of private actions, and hence the public version is not very different.

In Logistics, the PS-RTDP sends in many cases fewer than $\frac{1}{10}$ of the messages, and is hence considerably faster on many problems. In Logistics, PS-RTDP also required much fewer trajectories to converge. In Depot, as in blocks world, the number of messages sent by PS-RTDP is about one third of the messages sent by the complete DRTDP. In this domain, however, the approximate version required about half the time before converging, especially on larger problems.

5 Conclusion and Future Work

We presented two algorithms for CPPP in stochastic environments. DRTDP is a distributed, privacy preserving adaptation of RTDP, which is identical in execution to RTDP on the joint problem. PS-RTDP is an approximation of DRTDP that shares only public changes between agents to reduce the amount of messages during planning. Experiments show that PS-RTDP sends significantly less messages than DRTDP, sending an order of magnitude fewer messages in some cases. Future research can advance multiple trajectories simultaneously, allowing agents that do not currently advance a trajectory to start a new one. sequential action execution. We will investigate an extension to concurrent execution.

ea fuga nihil illo excepturi similique enim alias obcaecati voluptatum, laborum itaque qui saepe rerum?