8

Too Many cooks: Bayesian inference for coordinating Multi-agent Collaboration

Rose E. Wang[1], Sarah A. Wu[1], James A. Evans[2], David C. Parkes[3], Joshua B. Tenenbaum[1], and Max Kleiman-Weiner[1,3]

[1] *Massachusetts Institute of Technology,* [2] *University of Chicago, and*

[3] *Harvard University, USA*

8.1 Introduction

*Working together enables a group of agents to achieve together what no individual could achieve on their own (Tomasello, 2014; Henrich, 2015). However, collaboration is challenging as it requires agents to coordinate their behaviours. In the absence of prior experience, social roles, and norms, we still find ways to negotiate our joint behaviour in any given moment to work together with efficiency (Tomasello, Carpenter, Call, Behne and Moll, 2005; Misyak, Melkonyan, Zeitoun and Chater, 2014). Whether we are writing a scientific manuscript with collaborators or preparing a meal with friends, core questions we ask ourselves are: how can I help out the group? What should I work on next, and with whom should I do it with? Figuring out how to flexibly coordinate a collaborative endeavor is a fundamental challenge for any agent in a multi-agent world.

Central to this challenge is that agents' reasoning about what they should do in a multi-agent context depends on the future actions and intentions of others. When agents, like people, make independent decisions, these intentions are unobserved. Actions can reveal information about intentions, but predicting them is difficult because of uncertainty and ambiguity—multiple intentions can produce the same action. In humans, the ability to understand intentions from actions is called theory-of-mind (ToM). Humans rely on this ability to cooperate in coordinated ways, even in novel situations (Tomasello, Carpenter, Call, Behne and Moll, 2005; Shum, Kleiman-Weiner, Littman and Tenenbaum, 2019).

* Rose Wang and Sarah Wu contributed equally to this chapter

Rose E. Wang, Sarah A. Wu, James A. Evans, David C. Parkes, Joshua B. Tenenbaum, and Max Kleiman-Weiner, *Too Many cooks: Bayesian inference for coordinating Multi-agent Collaboration* In: *Human Like Machine Intelligence.* Edited by: Stephen Muggleton and Nick Charter, Oxford University Press. © Oxford University Press (2021). DOI: 10.1093/oso/9780198862536.003.0008

We aim to build agents with theory-of-mind and use these abilities for coordinating collaboration.

In this work, we study these abilities in the context of multiple agents cooking a meal together, inspired by the video game *Overcooked* (Ghost Town Games, 2016). These problems have hierarchically organized sub-tasks and share many features with other object-oriented tasks such as construction and assembly. These sub-tasks allow us to study agents that are challenged to coordinate in three distinct ways: (A) Divide and conquer: agents should work in parallel when sub-tasks can be efficiently carried out individually, (B) Cooperation: agents should work together on the same sub-task when most efficient or necessary, (C) Spatio-temporal movement: agents should avoid getting in each other's way at any time.

To illustrate, imagine the process required to make a simple salad: first chopping both tomato and lettuce and then assembling them together on a plate. Two people might collaborate by first dividing the sub-tasks up: one person chops the tomato and the other chops the lettuce. This doubles the efficiency of the pair by completing sub-tasks in parallel (challenge A). On the other hand, some sub-tasks may require multiple to work together. If only one person can use the knife and only the other can reach the tomatoes, then they must cooperate to chop the tomato (challenge B). In all cases, agents must coordinate their low-level actions in space and time to avoid interfering with others and be mutually responsive (challenge C).

Our work builds on a long history of using cooking tasks for evaluating multi-agent coordination across hierarchies of sub-tasks (Grosz and Kraus, 1996; Cohen and Levesque, 1991; Tambe, 1997). Most recently, environments inspired by *Overcooked* have been used in deep reinforcement learning studies where agents are trained using self-play and human data (Song, Wang, Lukasiewicz, Xu and Xu, 2019; Carroll, Shah, Ho, Griffiths, Seshia, Abbeel and Dragan, 2019). In contrast, our approach is based on techniques that dynamically learn while interacting rather than requiring large amounts of pre-training experience for a specific environment, team configuration, and sub-task structure. Instead our work shares goals with the ad-hoc coordination literature, where agents must adapt on the fly to variations in task, environment, or team (Chalkiadakis and Boutilier, 2003; Stone, Kaminka, Kraus and Rosenschein, 2010; Barrett, Stone and Kraus, 2011). However, prior work is often limited to action coordination (e.g,. chasing or hiding) rather than coordinating actions across and within sub-tasks. Our approach to this problem takes inspiration from the cognitive science of how people coordinate their cooperation in the absence of communication (Kleiman-Weiner, Ho, Austerweil, Littman and Tenenbaum, 2016). Specifically, we build on recent algorithmic progress in Bayesian theory-of-mind (Ramırez and Geffner, 2011; Nakahashi, Baker and Tenenbaum, 2016; Baker, Jara-Ettinger, Saxe and Tenenbaum, 2017; Shum, Kleiman-Weiner, Littman and Tenenbaum, 2019) and learning statistical models of others (Barrett, Stone, Kraus and Rosenfeld, 2012; Melo and Sardinha, 2016), and extend these works to decentralized multi-agent contexts.

Our strategy for multi-agent hierarchical planning builds on previous work linking high-level coordination (sub-tasks) to low-level navigation (actions) (Amato, Konidaris, Kaelbling and How, 2019). In contrast to models which have explicit communication mechanism or centralized controllers (McIntire, Nunes and Gini, 2016; Brunet, Choi and How, 2008), our approach is fully decentralized and our agents are never trained together. Prior work has also investigated ways in which multi-agent teams can mesh inconsistent plans (e.g. two agents doing the same sub-task by themselves) into consistent plans (e.g. the agents perform different sub-tasks in parallel) (Cox and Durfee, 2004, 2005), but these methods have also been centralized. We draw more closely from decentralized multi-agent planning approaches in which agents aggregate the effects of others and best respond (Claes, Robbel, Oliehoek, Tuyls, Hennes and Van der Hoek, 2015; Claes, Oliehoek, Baier and Tuyls, 2017). These prior works focus on tasks with spatial sub-tasks called *Spatial Task Allocation Problems* (SPATAPs). However, there are no mechanisms for agents to cooperate on the same sub-task as each sub-task is spatially distinct.

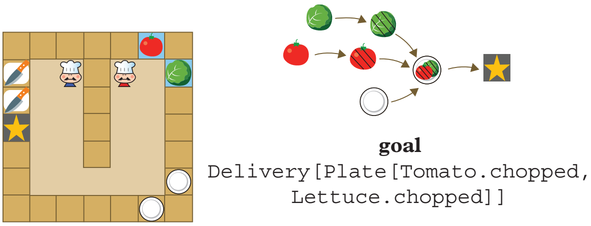
We develop *Bayesian Delegation*, an algorithm for decentralized multi-agent coordination that rises to the challenges described above. Bayesian Delegation leverages Bayesian inference with inverse planning to rapidly infer the sub-tasks others are working on. Our probabilistic approach allows agents to predict the intentions of other agents under uncertainty and ambiguity. These inferences allow agents to efficiently delegate their own efforts to the most high-value collaborative tasks for collective success. We quantitatively measure the performance of Bayesian Delegation in a suite of novel multi-agent environments. First, Bayesian Delegation outperforms existing approaches, completing all environments in less time than alternative approaches and maintaining performance even when scaled up to larger teams. Finally, we show Bayesian Delegation is an ad-hoc collaborator. It performs better than baselines when paired with alternative agents.

8.2 Multi-Agent MDPs with Sub-Tasks

A multi-agent Markov decision process (MMDP) with sub-tasks is described as a tuple $\langle n, \mathcal{S}, \mathcal{A}_{1\dots n}, T, R, \gamma, \mathcal{T} \rangle$ where n is the number of agents, $s \in \mathcal{S}$ are object-oriented states specified by the locations, status and type of each object and agent in the environment (Boutilier, 1996; Diuk, Cohen and Littman, 2008). $\mathcal{A}_{1\dots n}$ is the joint action space with $a_i \in \mathcal{A}_i$ being the set of actions available to agent i; each agent chooses its own actions independently. $T(s, a_{1\dots n}, s')$ is the transition function which describes the probability of transitioning from state s to s' after all agents act $a_{1\dots n}$. $R(s, a_{1\dots n})$ is the reward function shared by all agents and γ is the discount factor. Each agent aims to find a policy $\pi_i(s)$ that maximizes expected discounted reward. The environment state is fully observable to all agents, but agents do not observe the policies $\pi_{-i}(s)$ ($-i$ refers to all other agents except i) or any other internal representations of others agents.

Unlike traditional MMDPs, the environments we study have a partially ordered set of sub-tasks $\mathcal{T} = \{\mathcal{T}_0 \dots \mathcal{T}_{|\mathcal{T}|}\}$. Each sub-task $\mathcal{T}_i$ has preconditions that specify when a sub-task can be started, and postconditions that specify when it is completed. They provide structure when R is very sparse. These sub-tasks are also the target of high-level coordination between agents. In this work, all sub-tasks can be expressed as `Merge(X,Y)`, that is, to bring `X` and `Y` into the same location. Critically, unlike in SPATAPs, this location is not fixed or predetermined if both `X` and `Y` are movable. In the cooking environments we study here, the partial order of sub-tasks refers to a "recipe". Figure 8.1 shows an example of sub-task partial orders for a recipe.

The partial order of sub-tasks ($\mathcal{T}$) introduces two coordination challenges. First, `Merge` does not specify how to implement that sub-task in terms of efficient actions nor



```
Merge(Tomato.unchopped,Knife)
Merge(Lettuce.unchopped,Knife)
Merge(Tomato.chopped,
Lettuce.chopped)
Merge([Tomato.chopped
Lettuce.chopped],Plate[])
Merge(Plate[Tomato.chopped,
Lettuce.chopped],Delivery)
```

(a) Partial-Divider.

(b) Salad.

(c) Example sub-task order for Salad.

Figure 8.1 *The Overcooked environment. (a) The Partial-Divider kitchen offers many counters for objects, but forces agents to move through a narrow bottleneck. (b) The Salad recipe in which two chopped foods must be combined on a single plate and delivered, and (c) one of the many possible orderings for completing this task. All sub-tasks are expressed in the* `Merge` *operator. Different recipes are possible in each kitchen, allowing for variation in high-level goals while keeping the low-level navigation challenges fixed.*

which agent(s) should work on it. Second, because the ordering of sub-tasks is partial, the sub-tasks can be accomplished in many different orders. For instance, in the *Salad* recipe (Figure 8.1b), once the tomato and lettuce are chopped, they can: (a) first combine the lettuce and tomato and then plate, (b) the lettuce can be plated first and then add the tomato, or (c) the tomato can be plated first and then add the lettuce. These distinct orderings make coordination more challenging since to successfully coordinate, agents must align their ordering of sub-tasks.

The partially ordered set of sub-tasks $\mathcal{T}$ is given in the environment and generated by representing each recipe as an instance of STRIPS, an action language (Fikes and Nilsson, 1971). Each instance consists of an initial state, a specification of the goal state, and a set of actions with preconditions that dictate what must be true/false for the action to be executable, and postconditions that dictate what is made true/false when the action is executed. For instance, for the STRIPS instance of the recipe *Tomato*, the initial state is the initial configuration of the environment (i.e. all objects and their states), the specification of the goal state is `Delivery[Plate[Tomato.chopped]]`, and the actions are the `Merge` sub-tasks.. A plan for a STRIPS instance is a sequence of actions that can be executed from the initial state and results in a goal state. To generate these partial orderings, we construct a graph for each recipe in which the nodes are the states of the environment objects and the edges are valid actions. We then run breadth-first-search starting from the initial state to determine the nearest goal state, and return all shortest "recipe paths" between the two states.

8.2.1 Coordination Test Suite

We now describe the Overcooked inspired environments we use as a test suite for evaluating multi-agent collaboration. Each environment is a 2D grid-world kitchen. Figure 8.1a shows an example layout. The kitchens are built from counters that contain both movable food and plates and immovable stations (e.g. knife stations). The state is

Table 8.1 *State representation and transitions for the objects and interactions in the Overcooked environments. The two food items (tomato and lettuce) can be in either chopped or unchopped states. Objects with status [] are able to "hold" other objects. For example, an Agent holding a Plate holding an unchopped tomato would be denoted Agent[Plate[Tomato.unchopped]]. Once combined, these nested objects share the same $\{x, y\}$ coordinates and movement. Interaction dynamics occur when the two objects are in the same $\{x, y\}$ coordinates.*

Object state representation:

Type	Location	Status
Agent	{x, y}	[]
Plate	{x, y}	[]
Counter	{x, y}	[]
Delivery	{x, y}	[]
Knife	{x, y}	N/A
Tomato	{x, y}	{chopped, unchopped}
Lettuce	{x, y}	{chopped, unchopped}

Interaction dynamics:

Food.unchopped + Knife → Food.chopped + Knife

Food1 + Food2 → [Food1, Food2]

X + Y[] → Y[X]

represented as a list of entities and their type, location, and status (Diuk, Cohen and Littman, 2008). See Table 8.1 for a description of the different entities, the dynamics of object interactions, and the statuses that are possible. Agents (the chef characters) can move north, south, east, west or stay still. All agents move simultaneously. They cannot move through each other, into the same space, or through counters. If they try to do so, they remain in place instead. Agents pick up objects by moving into them and put down objects by moving into a counter while holding them. Agents chop foods by carrying the food to a knife station. Food can be merged with plates. Agents can only carry one object at a time and cannot directly pass to each other.

The goal in each environment is to cook a recipe in as few time steps as possible. The environment terminates after either the agents bring the finished dish specified by the recipe to the star square or 100 time steps elapse.

8.3 Bayesian Delegation

We now introduce *Bayesian Delegation*, a novel algorithm for multi-agent coordination that uses inverse planning to make probabilistic inferences about the sub-tasks other agents are performing. Bayesian Delegation models the latent intentions of others in order to dynamically decide whether to divide-and-conquer or to cooperate, and an action planner finds approximately optimal policies for each sub-task. Note that planning is decentralized at both levels, i.e., agents plan and learn for themselves without any access to each other's internal representations.

Inferring the sub-tasks others are working on enables each agent to select the right sub-task when multiple are possible. Agents maintain and update a belief state over the possible sub-tasks that all agents (including itself) are likely working on based on a history of observations that is commonly observed by all. Formally, Bayesian Delegation maintains a probability distribution over task allocations. Let $\mathbf{ta}$ be the set of all possible allocations of agents to sub-tasks where all agents are assigned to a sub-task. For example, if there are two sub-tasks ($[\mathcal{T}_1, \mathcal{T}_2]$) and two agents ($[i,j]$), then $\mathbf{ta} = [(i : \mathcal{T}_1, j : \mathcal{T}_2), (i : \mathcal{T}_2, j : \mathcal{T}_1), (i : \mathcal{T}_1, j : \mathcal{T}_1), (i : \mathcal{T}_2, j : \mathcal{T}_2)]$ where $i : \mathcal{T}_1$ means that agent i is "delegated" to sub-task $\mathcal{T}_1$. Thus, $\mathbf{ta}$ includes both the possibility that agents will divide and conquer (work on separate sub-tasks) and cooperate (work on shared sub-tasks). If all agents pick the same $ta \in \mathbf{ta}$, then they will easily coordinate. However, in our environments, agents cannot communicate before or during execution. Instead Bayesian Delegation maintains uncertainty about which ta the group is coordinating on, $P(ta)$.

At every time step, each agent selects the most likely allocation $ta^* = \arg\max_{ta} P(ta|H_{0:T})$, where $P(ta|H_{0:T})$ is the posterior over ta after having observed a history of actions $H_{0:T} = [(s_0, \mathbf{a_0}), \ldots (s_T, \mathbf{a}_T)]$ of T time steps and $\mathbf{a}_t$ are all agents' actions at time step t. The agent then plans the next best action according to ta^* using a model-based reinforcement learning algorithm described below. This posterior is computed according by Bayes rule:

$$P(ta|H_{0:T}) \propto P(ta)P(H_{0:T}|ta) = P(ta)\prod_{t=0}^{T} P(\mathbf{a_t}|s_t, ta) \tag{8.1}$$

where $P(ta)$ is the prior over ta and $P(\mathbf{a_t}|s_t, ta)$ is the likelihood of actions at time step t for all agents. Note that these belief updates do not explicitly consider the private knowledge that each agent has about their own intention at time $T-1$. Instead each agent performs inference based only on the history observed by all, i.e., the information a third-party observer would have access to (Sugden, 2003; Bacharach, 1999; Nagel, 1986). The likelihood of a given ta is the likelihood that each agent i is following their assigned task ($\mathcal{T}_i$) in that ta.

$$P(\mathbf{a_t}|s_t, ta) \propto \prod_{i:\mathcal{T} \in ta} exp(\beta * Q^*_{\mathcal{T}_i}(s, a_i)) \tag{8.2}$$

where $Q^*_{\mathcal{T}_i}(s, a_i)$, is the expected future reward of a towards the completion of sub-task $\mathcal{T}_i$ for agent i. The soft-max accounts for non-optimal and variable behavior as is typical in Bayesian theory-of-mind (Kleiman-Weiner, Ho, Austerweil, Littman and Tenenbaum, 2016; Baker, Jara-Ettinger, Saxe and Tenenbaum, 2017; Shum, Kleiman-

Weiner, Littman and Tenenbaum, 2019). β controls the degree to which an agent believes others are perfectly optimal. When $\beta \to 0$, the agent believes others are acting randomly. When $\beta \to \infty$, the agent believes others are perfectly maximizing. Since the likelihood is computed by planning, this approach to posterior inference is called inverse planning. Note that even though agents see the same history of states and actions, their belief updates will not necessarily be the same because updates come from $Q_{\mathcal{T}_i}$, which is computed independently for each agent and is affected by stochasticity in exploration.

The prior over $P(ta)$ is computed directly from the environment. First, $P(ta) = 0$ for all ta that have sub-tasks without satisfied preconditions. We set the remaining priors to $P(ta) \propto \sum_{\mathcal{T} \in ta} \frac{1}{V_{\mathcal{T}(s)}}$, where $V_{\mathcal{T}(s)}$ is the estimated value of the current state under sub-task $\mathcal{T}$. This gives ta that can be accomplished in less time a higher prior weight. Priors are reinitialized when new sub-tasks have their preconditions satisfied and when others are completed. Figure 8.2 shows an example of the dynamics of $P(ta)$ during agent interaction. The figure illustrates how Bayesian delegation enables agents to dynamically align their beliefs about who is doing what (i.e., assign high probability to a single ta).

Action planning transforms sub-task allocations into efficient actions and provides the critical likelihood for Bayesian Delegation (see Equation 8.1). Action planning takes the ta selected by Bayesian Delegation and outputs the next best action while modeling the movements of other agents. In this work, we use bounded real-time dynamic programming (BRTDP) extended to a multi-agent setting to find approximately optimal Q-values and policies (McMahan, Likhachev and Gordon, 2005). Each simulation was run in parallel (3 recipes, 3 environments, 5 models, 20 seeds) each on 1 CPU core, which took up to 15 GB of memory and roughly 3 hours to complete. We next describe the details of our BRTDP implementation: (McMahan, Likhachev and Gordon, 2005).

$$V^b_{\mathcal{T}_i}(s) = \min_{a \in \mathcal{A}_i} Q^b_{\mathcal{T}_i}(s,a), \quad V^b_{\mathcal{T}_i}(g) = 0$$
$$Q^b_{\mathcal{T}_i}(s,a) = C_{\mathcal{T}_i}(s,a) + \sum_{s' \in S} T(s'|s,a) V^b_{\mathcal{T}_i}(s')$$

where C is cost and $b = [l,u]$ is the lower and upper bound respectively. Each time step is penalized by 1 and movement (as opposed to staying still) by an additional 0.1. This cost structure incentivizes efficiency. The lower-bound was initialized to the Manhattan distance between objects (which ignores barriers). The upper-bound was the sum of the shortest-paths between objects which ignores the possibility of more efficiently passing objects. While BRTDP and these heuristics are useful for the specific spatial environments and subtask structures we develop here, it could be replaced with any other algorithm for finding an approximately optimal single-agent policy for a given sub-task. For details on how BRTDP updates on V and Q, see (McMahan, Likhachev and Gordon, 2005). BRTDP was run until the bounds converged ($\alpha = 0.01, \tau = 2$) or for a maximum of 100 trajectories each with up to 75 roll-outs for all models. The softmax during inference used $\beta = 1.3$. At each time step, agents select the action with the highest value for their sub-task. When agents do not have any valid sub-tasks, i.e. sub-task is None, they take a random action (uniform across the movement and

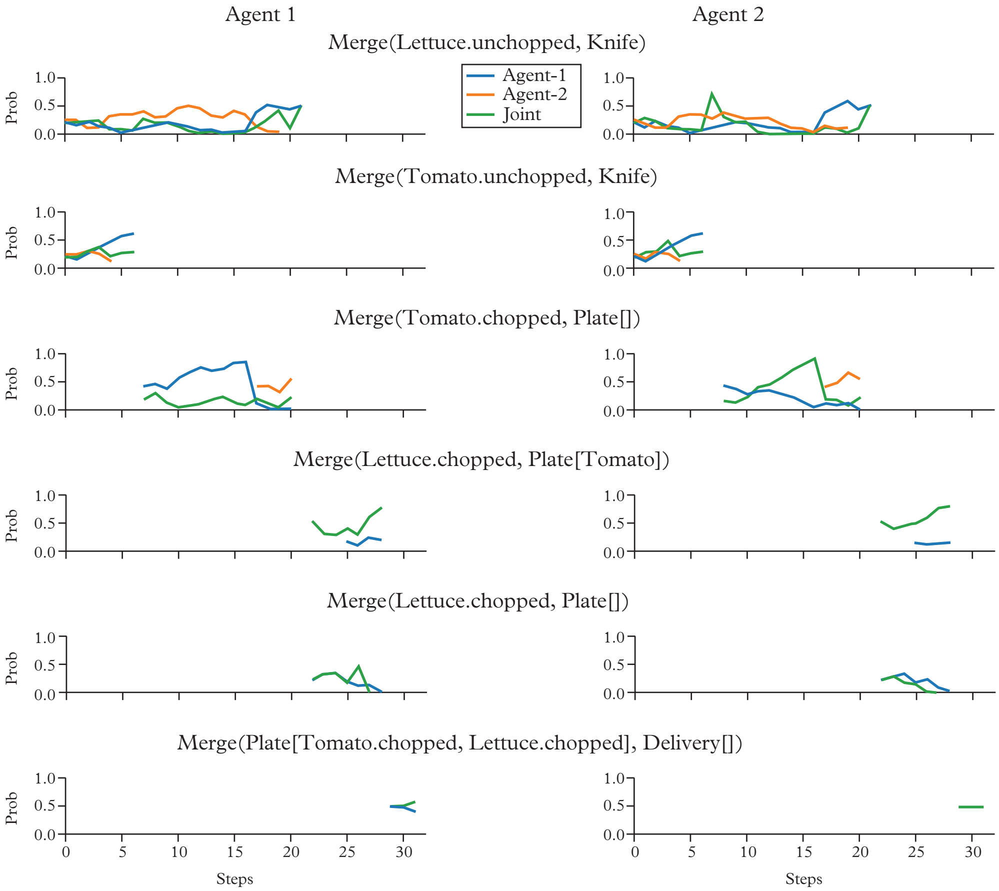


Figure 8.2 *Dynamics of the belief state, P(ta) for each agent during Bayesian delegation with the Salad recipe on Partial-Divider (Figure 8.1). During the first 7 time steps, only the* `Merge(Lettuce.unchopped, Knife)` *and* `Merge(Tomato.unchopped, Knife)` *sub-tasks are nonzero because their preconditions are met. These beliefs show alignment across the ordering of sub-tasks as well as within each sub-task. Salad can be completed in three different ways (see Figure 8.5), yet both agents eventually drop* `Merge(Lettuce.unchopped, Plate[])` *in favor of* `Merge(Tomato.unchopped, Plate[])` *followed by* `Merge(Lettuce.chopped, Plate[Tomato])`*. Agents' beliefs also converge over the course of each specific sub-task. For instance, while both agents are at first uncertain about who should be delegated to* `Merge(Lettuce.unchopped, Knife)`*, they eventually align to the same relative ordering. This alignment continues, even though there is never any communication or prior agreement on what sub-task each agent should be doing or when.*

stay-in-place actions). This greatly improves the performance of the alternative (lesioned) models: without this noise, they often get stuck and block each other from completing the recipe. It has no effect on Bayesian Delegation.

Agents use ta^* from Bayesian Delegation to address two types of low-level coordination problems: (1) avoiding getting in each others way while working on distinct sub-tasks, and (2) cooperating efficiently when working on a shared sub-task. ta^* contains agent i's best guess about the sub-tasks carried out by others, $\mathcal{T}_{-i}$. In the first case, $\mathcal{T}_i \neq \mathcal{T}_{-i}$. Agent i first creates models of the others performing $\mathcal{T}_{-i}$ assuming others agents are stationary ($\pi^0_{\mathcal{T}_{-i}}(s)$, level-0 models). These level-0 models are used to reduce the multi-agent transition function to a single agent transition function T' where the transitions of the other agents are assumed to follow the level-0 policies, $T'(s'|s, a_{-i}) = \sum_{a_i} T(s'|s, a_{-i}, a_i) \prod_{A \in -i} \pi^0_{\mathcal{T}_A}(s)$. Running BRTDP on this transformed environment finds an approximately optimal level-1 policy $\pi^1_{\mathcal{T}_i}(s)$ for agent i that "best responds" to the level-0 models of the other agents. This approach is similar to level-K or cognitive hierarchy (Wright and Leyton-Brown, 2010; Kleiman-Weiner, Ho, Austerweil, Littman and Tenenbaum, 2016; Shum, Kleiman-Weiner, Littman and Tenenbaum, 2019).

When $\mathcal{T}_i = \mathcal{T}_{-i}$, agent i attempts to work together on the same sub-task with the other agent(s). The agent simulates a fictitious centralized planner that controls the actions of all agents working together on the same sub-task (Kleiman-Weiner, Ho, Austerweil, Littman and Tenenbaum, 2016). This transforms the action space: if both i and j are working on $\mathcal{T}_i$, then $\mathcal{A}' = a_i \times a_j$. Joint policies $\pi^J_{\mathcal{T}_i}(s)$ can similarly be found by single-agent planners such as BRTDP. Agent i then takes the actions assigned to it under $\pi^J_{\mathcal{T}_i}(s)$. Joint policies enable emergent decentralized cooperative behavior—agents can discover efficient and novel ways of solving sub-tasks as a team such as passing objects across counters. Since each agent is solving for their own $\pi^J_{\mathcal{T}_i}(s)$, these joint policies are not guaranteed to be perfectly coordinated due to stochasticity in the planning process. Note that although we use BRTDP, any other model-based reinforcement learner or planner could also be used.

8.4 Results

We evaluate the performance of Bayesian Delegation across two different experimental paradigms. First, we test the performance of each agent type when all agents are the same type with both two and three agents (self-play). Second, we test the performance of each agent type when paired with an agent of a different type (ad-hoc).

We compare the performance of Bayesian Delegation (BD) to four alternative baseline agents: Uniform Priors (UP), which starts with uniform probability mass over all valid ta and updates through inverse planning; Fixed Beliefs (FB), which does not update $P(ta)$ in response to the behavior of others; Divide and Conquer (D&C) (Ephrati and Rosenschein, 1994), which sets $P(ta) = 0$ if that ta assigns two agents to the same sub-task (this is conceptually similar to Empathy by Fixed Weight Discounting (Claes, Robbel, Oliehoek, Tuyls, Hennes and Van der Hoek, 2015) because agents cannot share sub-tasks and D&C discounts sub-tasks most likely to be attended to by other agents

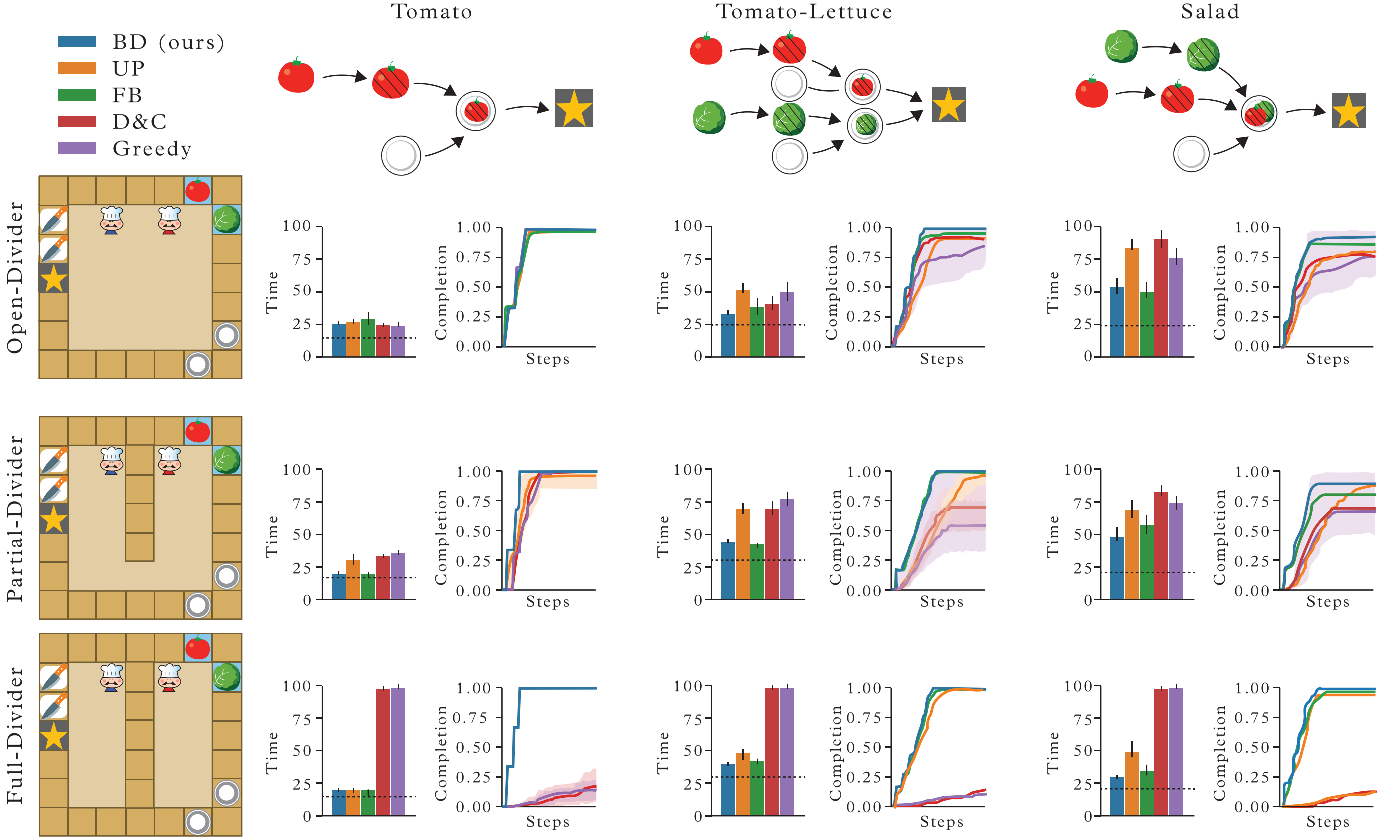


Figure 8.3 *Performance results for each kitchen-recipe composition (lower is better) for two agents in self-play. The row shows the kitchen and the column shows the recipe. Within each composition, the left graph shows the number of time steps needed to complete all sub-tasks. The dashed lines on the left graph represent the optimal performance of a centralized team. The right graph shows the fraction of sub-tasks completed over time. Bayesian Delegation completes more sub-tasks and does so more quickly compared to baselines.*

based on $P(ta|H)$); Greedy, which selects the sub-task it can complete most quickly without considering the sub-tasks other agents are working on. All agents take advantage of the sub-task structure because end-to-end optimization of the full recipe using techniques such as DQN (Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra and Riedmiller, 2013) and Q-learning (Watkins and Dayan, 1992) never succeeded under our computational budget.

To highlight the differences between our model and the alternatives, let us consider an example with two possible sub-tasks ($[\mathcal{T}_1, \mathcal{T}_2]$) and two agents ($[i, j]$). The prior for Bayesian Delegation puts positive probability mass on $\mathbf{ta} = [(i : \mathcal{T}_1, j : \mathcal{T}_2), (i : \mathcal{T}_2, j : \mathcal{T}_1), (i : \mathcal{T}_1, j : \mathcal{T}_1), (i : \mathcal{T}_2, j : \mathcal{T}_2)]$ where $i : \mathcal{T}_1$ means that agent i is assigned to sub-task $\mathcal{T}_1$. The UP agent proposes the same $\mathbf{ta}$, but places uniform probability across all elements, i.e., $P(ta) = \frac{1}{4}$ for all $ta \in \mathbf{ta}$. FB would propose the same $\mathbf{ta}$ with the same priors as Bayesian Delegation, but would never update its beliefs. The D&C agent does not allow for joint sub-tasks, so it would reduce to $\mathbf{ta} = [(i : \mathcal{T}_1, j : \mathcal{T}_2), (i : \mathcal{T}_2, j : \mathcal{T}_1)]$. Lastly, Greedy makes no inferences so each agent i would propose $\mathbf{ta} = [(i : \mathcal{T}_1), (i : \mathcal{T}_2)]$. Note that j does not appear.

In the first two computational experiments, we analyze the results in terms of three key metrics. The two pivotal metrics are the number of time steps to complete the full recipe and the total fraction of sub-tasks completed. We also analyze average number of shuffles, a measure of uncoordinated behavior. A *shuffle* is any action that negates the previous action, such as moving left and then right, or picking an object up and then putting it back down (see Figure 8.4a for an example). All experiments show the average performance over 20 random seeds. Agents are evaluated in 9 task-environment combinations (3 recipes × 3 kitchens).

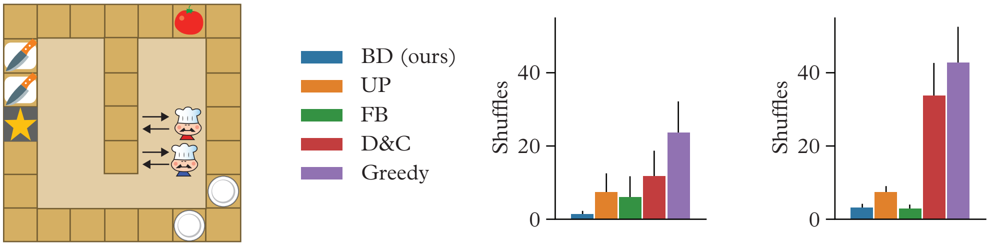
8.4.1 Self-play

Table 8.2 quantifies the performance of all agents aggregated across the 9 environments. Bayesian Delegation outperforms all baselines and completes recipes with less time step and fewer shuffles. The performance gap was even larger with three agents. Most other agents performed worse with three agents than they did with two, while the performance of Bayesian Delegation did not suffer. Figure 8.3 breaks down performance by kitchen and recipe. All five types of agents are comparable when given the recipe *Tomato* in *Open-Divider*, but when faced with more complex situations, Bayesian Delegation outperform the others. For example, without the ability to represent shared sub-tasks, D&C and Greedy fail in *Full-Divider* because they cannot explicitly coordinate on the same sub-task to pass objects across the counters. Baseline agents were also less capable of low-level coordination resulting in more inefficient shuffles (Figure 8.4). A breakdown of three agent performance is shown in Figure 8.7.

Learning about other agents is especially important for more complicated recipes that can be completed in different orders. In particular, FB and Greedy, which do not learn, have trouble with the *Salad* recipe on *Full Divider*. There are two challenges in this composition. One is that the *Salad* recipe can be completed in three different orders: once the tomato and lettuce are chopped, they can be (a) first combined together and

Table 8.2 *Self-play performance of our model and alternative models with two versus three agents. All metrics are described in the text. See Figure 8.3 for more detailed results on Time Steps and Completion for two agents in self-play, and see Figure 8.4 for more detailed results on shuffles. Averages ± standard error of the mean.*

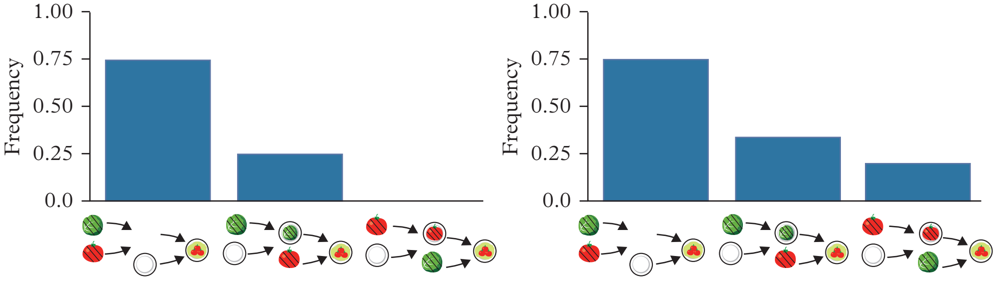
		Time Steps	Completion	Shuffles
		(↓ better)	(↑ better)	(↓ better)
Two agents	BD (ours)	**35.29** ± **1.40**	**0.98** ± **0.06**	**1.01** ± **0.05**
	UP	50.42 ± 2.04	0.94 ± 0.05	5.32 ± 0.03
	FB	37.58 ± 1.60	0.95 ± 0.04	2.64 ± 0.03
	D&C	71.57 ± 2.40	0.61 ± 0.07	13.08 ± 0.05
	Greedy	71.11 ± 2.41	0.57 ± 0.08	17.17 ± 0.06
Three agents	BD (ours)	**34.52** ± **1.66**	**0.96** ± **0.08**	1.64 ± 0.05
	UP	56.84 ± 2.12	0.91 ± 0.22	5.02 ± 0.12
	FB	41.34 ± 2.27	0.92 ± 0.08	**1.55** ± **0.05**
	D&C	67.21 ± 2.31	0.67 ± 0.15	4.94 ± 0.09
	Greedy	75.87 ± 2.32	0.62 ± 0.22	12.04 ± 0.13



(a) Example "shuffle" (b) *Open-Divider* (c) *Partial-Divider*

Figure 8.4 *Shuffles observed for recipe Tomato+Lettuce. (a) Example of a shuffle, where both agents simultaneously move back and forth from left to right, over and over again. This coordination failure prevents them from passing each other. Note that they are not colliding. Average number of shuffles by each agent in the (b) Open-Divider and (c) Partial-Divider environments. Error bars show the standard error of the mean. Bayesian Delegation and Joint Planning help prevent shuffles, leading to better coordinated behavior.*

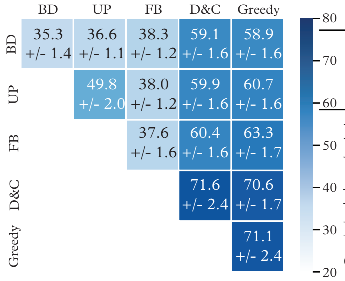
then plated, (b) the lettuce can be plated first and then the tomato added or (c) the tomato can be plated first and then the lettuce added. The second challenge is that neither agent can perform all the sub-tasks by themselves, thus they must converge to the same order. Unless the agents that do not learn coordinate by luck, they have no way of recovering. Figure 8.5 shows the diversity of orderings used across different runs of Bayesian Delegation. Another failure mode for agents lacking learning is that FB and Greedy frequently get stuck in cycles in which both agents are holding objects that must be merged (e.g., a plate and lettuce). They fail to coordinate their actions such that one puts their object down in order for the other to pick it up and merge. Bayesian



(a) *Open-Divider*

(b) *Partial-Divider*

Figure 8.5 *Frequency that three orderings of Salad are completed by our model agents, in (a) Open-Divider and (b) Partial-Divider.*



	BD	UP	FB	D&C	Greedy
BD	35.3 +/- 1.4	36.6 +/- 1.1	38.3 +/- 1.2	59.1 +/- 1.6	58.9 +/- 1.6
UP		49.8 +/- 2.0	38.0 +/- 1.2	59.9 +/- 1.6	60.7 +/- 1.6
FB			37.6 +/- 1.6	60.4 +/- 1.6	63.3 +/- 1.7
D&C				71.6 +/- 2.4	70.6 +/- 1.7
Greedy					71.1 +/- 2.4

	Times Steps (↓ better)	Completion (↑ better)	Shuffles (↓ better)
BD (ours)	**48.25 ± 0.75**	**0.90 ± 0.01**	**3.96 ± 0.33**
UP	48.84 ± 0.77	0.89 ± 0.01	4.17 ± 0.34
FB	50.00 ± 0.78	0.87 ± 0.01	5.11 ± 0.42
D&C	62.49 ± 0.83	0.77 ± 0.01	6.84 ± 0.43
Greedy	63.40 ± 0.84	0.76 ± 0.01	6.61 ± 0.41

Figure 8.6 *Ad-hoc performance of different agent pairs in time steps (the lower and lighter, the better). (Left) Rows and columns correspond to different agents. Each cell is the average performance of one the row agent playing with the column agent. (Right) Mean performance (± SE) of agents when paired with the others.*

Delegation can break these symmetries by yielding to others so long as they make net progress towards the completion of one of the sub-tasks. For these reasons, only Bayesian Delegation performs on par (if not more efficiently) with three agents than with two agents. As additional agents join the team, aligning plans becomes even more important in order for agents to avoid performing conflicting or redundant sub-tasks.

8.4.2 Ad-hoc

Next, we evaluated the ad-hoc performance of the agents. We show that Bayesian Delegation is a successful ad-hoc collaborator. Each agent was paired with the other agent types. None of the agents had any prior experience with the other agents. Figure 8.6 shows the performance of each agent when matched with each other and in aggregate across all recipe-kitchen combinations. Bayesian Delegation performed well even when matched with baselines. When paired with UP, D&C, and Greedy, the dyad performed better than when UP, D&C, and Greedy were each paired with their own type. Because

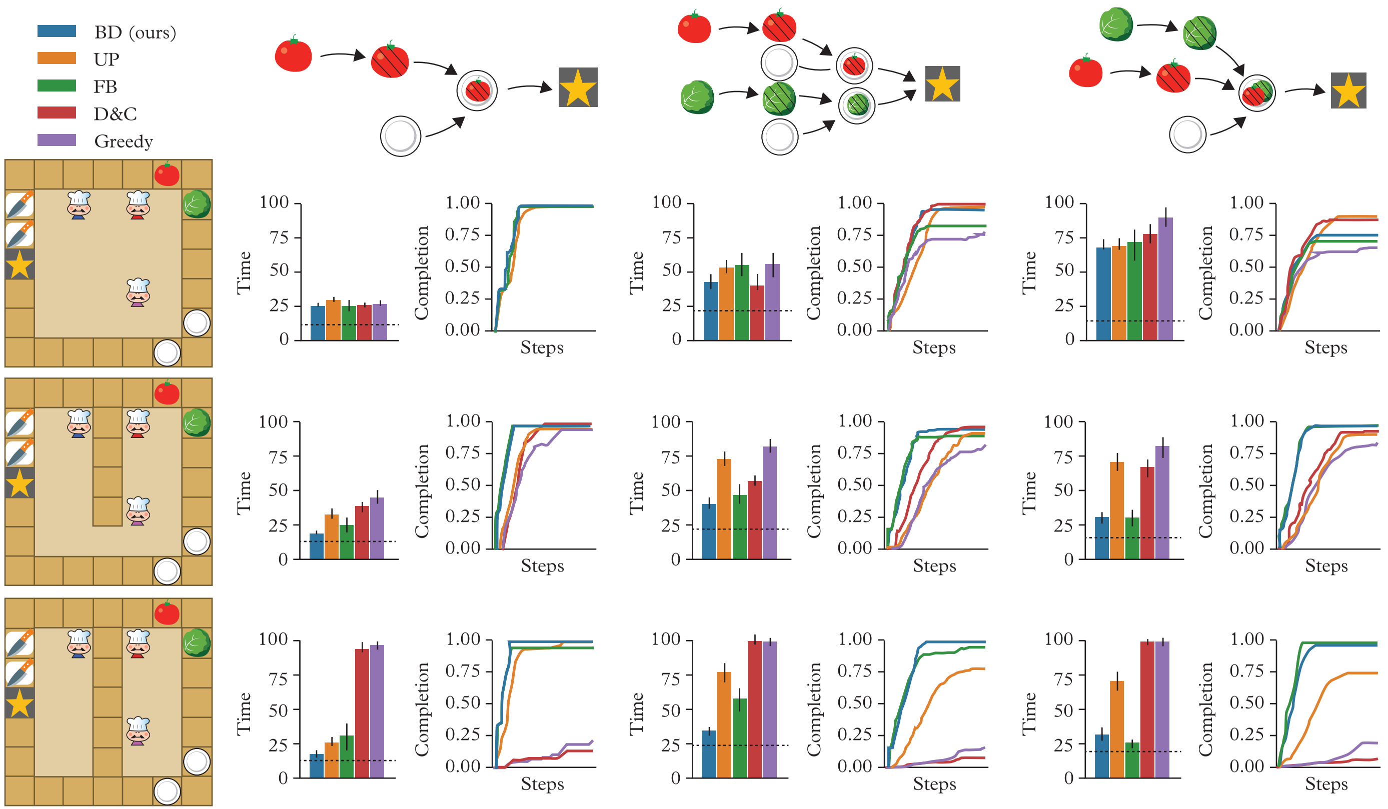
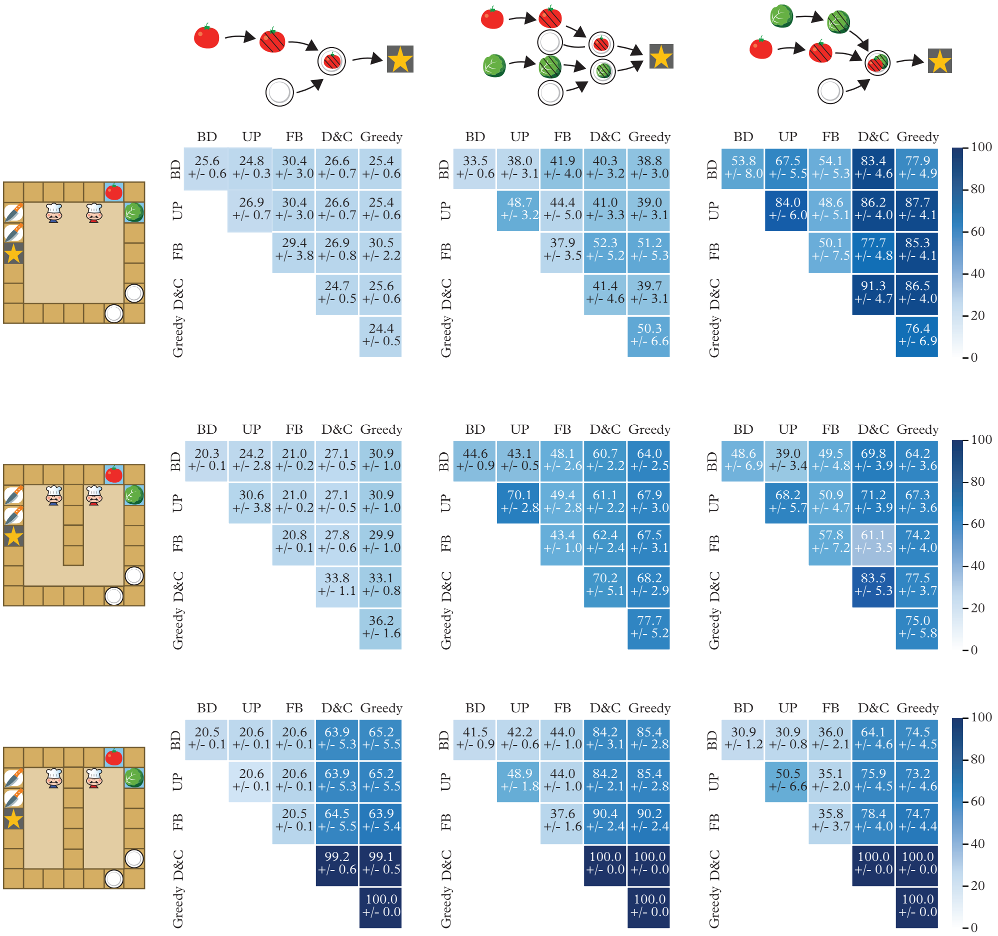


Figure 8.7 *Performance results for each kitchen-recipe composition (lower is better) for* ***three*** *agents. The row shows the kitchen and the column shows the recipe. Within each composition, the left graph shows the number of time steps needed to complete all sub-tasks. The dashed lines on the left graph represent the optimal performance of a centralized team. The right graph shows the fraction of sub-tasks completed over time. The full agent completes more sub-tasks and does so more quickly compared to the alternatives.*



	BD	UP	FB	D&C	Greedy
BD	25.6 +/- 0.6	24.8 +/- 0.3	30.4 +/- 3.0	26.6 +/- 0.7	25.4 +/- 0.6
UP		26.9 +/- 0.7	30.4 +/- 3.0	26.6 +/- 0.7	25.4 +/- 0.6
FB			29.4 +/- 3.8	26.9 +/- 0.8	30.5 +/- 2.2
D&C				24.7 +/- 0.5	25.6 +/- 0.6
Greedy					24.4 +/- 0.5

	BD	UP	FB	D&C	Greedy
BD	33.5 +/- 0.6	38.0 +/- 3.1	41.9 +/- 4.0	40.3 +/- 3.2	38.8 +/- 3.0
UP		48.7 +/- 3.2	44.4 +/- 5.0	41.0 +/- 3.3	39.0 +/- 3.1
FB			37.9 +/- 3.5	52.3 +/- 5.2	51.2 +/- 5.3
D&C				41.4 +/- 4.6	39.7 +/- 3.1
Greedy					50.3 +/- 6.6

	BD	UP	FB	D&C	Greedy
BD	53.8 +/- 8.0	67.5 +/- 5.5	54.1 +/- 5.3	83.4 +/- 4.6	77.9 +/- 4.9
UP		84.0 +/- 6.0	48.6 +/- 5.1	86.2 +/- 4.0	87.7 +/- 4.1
FB			50.1 +/- 7.5	77.7 +/- 4.8	85.3 +/- 4.1
D&C				91.3 +/- 4.7	86.5 +/- 4.0
Greedy					76.4 +/- 6.9

	BD	UP	FB	D&C	Greedy
BD	20.3 +/- 0.1	24.2 +/- 2.8	21.0 +/- 0.2	27.1 +/- 0.5	30.9 +/- 1.0
UP		30.6 +/- 3.8	21.0 +/- 0.2	27.1 +/- 0.5	30.9 +/- 1.0
FB			20.8 +/- 0.1	27.8 +/- 0.6	29.9 +/- 1.0
D&C				33.8 +/- 1.1	33.1 +/- 0.8
Greedy					36.2 +/- 1.6

	BD	UP	FB	D&C	Greedy
BD	44.6 +/- 0.9	43.1 +/- 0.5	48.1 +/- 2.6	60.7 +/- 2.2	64.0 +/- 2.5
UP		70.1 +/- 2.8	49.4 +/- 2.8	61.1 +/- 2.2	67.9 +/- 3.0
FB			43.4 +/- 1.0	62.4 +/- 2.4	67.5 +/- 3.1
D&C				70.2 +/- 5.1	68.2 +/- 2.9
Greedy					77.7 +/- 5.2

	BD	UP	FB	D&C	Greedy
BD	48.6 +/- 6.9	39.0 +/- 3.4	49.5 +/- 4.8	69.8 +/- 3.9	64.2 +/- 3.6
UP		68.2 +/- 5.7	50.9 +/- 4.7	71.2 +/- 3.9	67.3 +/- 3.6
FB			57.8 +/- 7.2	61.1 +/- 3.5	74.2 +/- 4.0
D&C				83.5 +/- 5.3	77.5 +/- 3.7
Greedy					75.0 +/- 5.8

	BD	UP	FB	D&C	Greedy
BD	20.5 +/- 0.1	20.6 +/- 0.1	20.6 +/- 0.1	63.9 +/- 5.3	65.2 +/- 5.5
UP		20.6 +/- 0.1	20.6 +/- 0.1	63.9 +/- 5.3	65.2 +/- 5.5
FB			20.5 +/- 0.1	64.5 +/- 5.5	63.9 +/- 5.4
D&C				99.2 +/- 0.6	99.1 +/- 0.5
Greedy					100.0 +/- 0.0

	BD	UP	FB	D&C	Greedy
BD	41.5 +/- 0.9	42.2 +/- 0.6	44.0 +/- 1.0	84.2 +/- 3.1	85.4 +/- 2.8
UP		48.9 +/- 1.8	44.0 +/- 1.0	84.2 +/- 2.1	85.4 +/- 2.8
FB			37.6 +/- 1.6	90.4 +/- 2.4	90.2 +/- 2.4
D&C				100.0 +/- 0.0	100.0 +/- 0.0
Greedy					100.0 +/- 0.0

	BD	UP	FB	D&C	Greedy
BD	30.9 +/- 1.2	30.9 +/- 0.8	36.0 +/- 2.1	64.1 +/- 4.6	74.5 +/- 4.5
UP		50.5 +/- 6.6	35.1 +/- 2.0	75.9 +/- 4.5	73.2 +/- 4.6
FB			35.8 +/- 3.7	78.4 +/- 4.0	74.7 +/- 4.4
D&C				100.0 +/- 0.0	100.0 +/- 0.0
Greedy					100.0 +/- 0.0

Figure 8.8 *Heat map performance for each kitchen-recipe composition (lower is better) for two agents using different models. The figure breaks down the aggregate performance from Figure 8.6 by kitchen (row) and recipe(column). In most compositions, as models perform as better adhoc coordinators as they become more "Bayesian Delegation"-like (going bottom to top by row, right to left by column).*

Bayesian Delegation can learn in-the-moment, it can overcome some of the ways that these agents get stuck. UP performs better when paired with Bayesian Delegation or FB compared to self-play, suggesting that as long as one of the agents is initialized with smart priors, it may be enough to compensate for the other's uninformed priors. D&C and Greedy perform better when paired with Bayesian Delegation, FB, or UP. Crucially, these three agents all represent cooperative plans where both agents cooperate on the same sub-task. Figure 8.8 breaks down the ad-hoc performance of each agent pairing by recipe and kitchen.

8.5 Discussion

We developed Bayesian Delegation, a new algorithm inspired by and consistent with human theory-of-mind. Bayesian Delegation enables efficient ad-hoc coordination by rapidly inferring the sub-tasks of others. Agents dynamically align their beliefs about who is doing what and determine when they should help another agent on the same sub-task and when they should divide and conquer for increased efficiency. It also enables them to complete sub-tasks that neither agent could achieve on its own. Our agents reflect many natural aspects of human cooperation, such as the emergence of joint behavior when joint planning is deemed better than planning alone (Tomasello, 2014).

The environments studied here are highly challenging from a coordination perspective. There are multiple ways to complete each goal and spatial movement is relatively constrained leading to a high probability of miscoordination. Furthermore, there are no channels for communication. If communication were possible in these environments, many of these coordination problems could be reasoned about directly. Instead, Bayesian Delegation is a kind of implicit mechanism for coordinating group behavior. One might hypothesize that implicit coordination mechanisms such as Bayesian Delegation were important for collaborative hunting and other kinds of early coordinated behavior (Tomasello, 2014). Indeed, these kinds of implicit, pre-linguistic mechanisms for coordinating the mental states of other may have been important for the emergence and acquisition of language (Misyak, Melkonyan, Zeitoun and Chater, 2014).

While Bayesian Delegation reflects progress towards human-like coordination, there are still limitations which we hope to address in future work. One challenge is that when agents jointly plan for a single sub-task, they currently have no way of knowing when they have completed their individual "part" of the joint effort. Consider a case where one agent needs to pass lettuce and tomato across the divider for the other to chop it, after dropping off the lettuce, the first agent is currently unable to reason that it has fulfilled its role in that joint plan and can move on, i.e., that the rest of the sub-task depends only on the actions of the other agent. Currently, our agents considers sub-tasks active as long as their post-conditions remain unsatisfied. If agents were able to recognize when their sub-tasks were finished with respect to themselves, then they would be able to coordinate even more efficiently and flexibly. This opens the possibility of looking ahead to future sub-tasks that will need to be done even before their preconditions are satisfied. For example, once an agent passes off a tomato to another to chop, the first agent can go and get a plate in anticipation of also passing that over even before the chopping has begun.

At some point, as one scales up the number of agents, there can be "too many cooks" in the kitchen! The algorithms presented here scale poorly with the number of agents. In some sense this is a natural trade-off, as Bayesian Delegation through inverse planning requires computing policies not just for oneself but also for each other agent. Other less flexible but more efficient mechanisms may also play a crucial role. Over time, people build up and establish behavioral norms and conventions which yield coordination without sophisticated agent modeling (Young, 1993; Bicchieri, 2006; Lewis, 1969). Roles often emerge between people that spend significant time together (Misyak, Melkonyan, Zeitoun and Chater, 2014). For instance in *Partial-Divider* a pair of agents could break the symmetry by converging on a norm where one person always yields to the other or in *Open-Divider* a pair of agents might decide to always move in a clockwise direction to minimize the probability of collisions (Lerer and Peysakhovich, 2019; Carroll, Shah, Ho, Griffiths, Seshia, Abbeel and Dragan, 2019). Models that allow for these kinds of subtle norms and roles to emerge are needed for agents to form longer term collaborations that persist beyond a single short interaction. Such representations are essential for building AI agents that are capable of partnering with human teams and with each other.

Acknowledgements

We thank Alex Peysakhovich, Barbara Grosz, DJ Strouse, Leslie Kaelbling, Micah Carroll, and Natasha Jaques for insightful ideas and comments. This work was funded by the Harvard Data Science Initiate, Harvard CRCS, Templeton World Charity Foundation, The Future of Life Institute, DARPA Ground Truth, and The Center for Brains, Minds and Machines (NSF STC award CCF-1231216).

References

Amato, Christopher, Konidaris, George, Kaelbling, Leslie Pack, and How, Jonathan P. (2019). Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research*, **64**, 817–859.

Bacharach, Michael (1999). Interactive team reasoning: A contribution to the theory of co-operation. *Research in economics*, **53**(2), 117–147.

Baker, Chris L, Jara-Ettinger, Julian, Saxe, Rebecca, and Tenenbaum, Joshua B (2017). Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, **1**, 0064.

Barrett, Samuel, Stone, Peter, and Kraus, Sarit (2011). Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 567–574. International Foundation for Autonomous Agents and Multiagent Systems.

Barrett, Samuel, Stone, Peter, Kraus, Sarit, and Rosenfeld, Avi (2012). Learning teammate models for ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*, pp. 57–63.

Bicchieri, Cristina (2006). *The grammar of society: The nature and dynamics of social norms*. Cambridge University Press.

Boutilier, Craig (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pp. 195–210. Morgan Kaufmann Publishers Inc.

Brunet, Luc, Choi, Han-Lim, and How, Jonathan (2008). Consensus-based auction approaches for decentralized task assignment. In *AIAA guidance, navigation and control conference and exhibit*, p. 6839.

Carroll, Micah, Shah, Rohin, Ho, Mark, Griffiths, Thomas, Seshia, Sanjit, Abbeel, Pieter, and Dragan, Anca (2019). On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems*.

Chalkiadakis, Georgios and Boutilier, Craig (2003). Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 709–716.

Claes, Daniel, Oliehoek, Frans, Baier, Hendrik, and Tuyls, Karl (2017). Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 492–500. International Foundation for Autonomous Agents and Multiagent Systems.

Claes, Daniel, Robbel, Philipp, Oliehoek, Frans A, Tuyls, Karl, Hennes, Daniel, and Van der Hoek, Wiebe (2015). Effective approximations for multi-robot coordination in spatially distributed tasks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 881–890. International Foundation for Autonomous Agents and Multiagent Systems.

Cohen, Philip R and Levesque, Hector J (1991). Teamwork. *Noûs*, **25**(4), 487–512.

Cox, Jeffrey S and Durfee, Edmund H (2004). Efficient mechanisms for multiagent plan merging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pp. 1342–1343. IEEE.

Cox, Jeffrey S and Durfee, Edmund H (2005). An efficient algorithm for multiagent plan coordination. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 828–835.

Diuk, Carlos, Cohen, Andre, and Littman, Michael L (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 240–247. ACM.

Ephrati, Eithan and Rosenschein, Jeffrey S (1994). Divide and conquer in multi-agent planning. In *AAAI*, Volume 1, p. 80.

Fikes, Richard E. and Nilsson, Nils J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, **2**(3), 189 – 208.

Ghost Town Games (2016). Overcooked.

Grosz, Barbara J and Kraus, Sarit (1996). Collaborative plans for complex group action. *Artificial Intelligence*, **86**(2), 269–357.

Henrich, Joseph (2015). *The secret of our success: how culture is driving human evolution, domesticating our species, and making us smarter*. Princeton University Press.

Kleiman-Weiner, Max, Ho, Mark K, Austerweil, Joseph L, Littman, Michael L, and Tenenbaum, Joshua B (2016). Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*.

Lerer, Adam and Peysakhovich, Alexander (2019). Learning existing social conventions via observationally augmented self-play. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 107–114. ACM.

Lewis, David (1969). *Convention: A philosophical study*. John Wiley & Sons.

McIntire, Mitchell, Nunes, Ernesto, and Gini, Maria (2016). Iterated multi-robot auctions for precedence-constrained task scheduling. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1078–1086.

McMahan, H Brendan, Likhachev, Maxim, and Gordon, Geoffrey J (2005). Bounded real-time dynamic programming: Rtdp with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd international conference on Machine learning*, pp. 569–576. ACM.

Melo, Francisco S and Sardinha, Alberto (2016). Ad hoc teamwork by learning teammates' task. *Autonomous Agents and Multi-Agent Systems*, **30**(2), 175–219.

Misyak, Jennifer B, Melkonyan, Tigran, Zeitoun, Hossam, and Chater, Nick (2014). Unwritten rules: virtual bargaining underpins social interaction, culture, and society. *Trends in cognitive sciences*.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nagel, Thomas (1986). *The view from nowhere*. Oxford University Press.

Nakahashi, Ryo, Baker, Chris L, and Tenenbaum, Joshua B (2016). Modeling human understanding of complex intentional action with a bayesian nonparametric subgoal model. In *AAAI*, pp. 3754–3760.

Ramırez, Miquel and Geffner, Hector (2011). Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *IJCAI*, pp. 2009–2014. IJCAI/AAAI.

Shum, Michael, Kleiman-Weiner, Max, Littman, Michael L, and Tenenbaum, Joshua B (2019). Theory of minds: Understanding behavior in groups through inverse planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

Song, Yuhang, Wang, Jianyi, Lukasiewicz, Thomas, Xu, Zhenghua, and Xu, Mai (2019). Diversity-driven extensible hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 33, pp. 4992–4999.

Stone, Peter, Kaminka, Gal A, Kraus, Sarit, and Rosenschein, Jeffrey S (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Sugden, Robert (2003). The logic of team reasoning. *Philosophical explorations*, **6**(3), 165–181.

Tambe, Milind (1997). Towards flexible teamwork. *Journal of artificial intelligence research*, 7, 83–124.

Tomasello, Michael (2014). *A natural history of human thinking*. Harvard University Press.

Tomasello, Michael, Carpenter, Malinda, Call, Josep, Behne, Tanya, and Moll, Henrike (2005). Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and brain sciences*, **28**(05), 675–691.

Watkins, Christopher JCH and Dayan, Peter (1992). Q-learning. *Machine learning*, 8(3-4), 279–292.

Wright, James R and Leyton-Brown, Kevin (2010). Beyond equilibrium: Predicting human behavior in normal-form games. In *AAAI*.

Young, H Peyton (1993). The evolution of conventions. *Econometrica: Journal of the Econometric Society*, 57–84.