Too many cooks: Bayesian inference for coordinating multi-agent collaboration

Sarah A. Wu* (sarahawu@alum.mit.edu)

Massachusetts Institute of Technology

Rose E. Wang* (rewang@alum.mit.edu)

Massachusetts Institute of Technology

James A. Evans (jevans@uchicago.edu)

University of Chicago

Joshua B. Tenenbaum (jbt@mit.edu)

Massachusetts Institute of Technology

David C. Parkes (parkes@eecs.harvard.edu)

Harvard University

Max Kleiman-Weiner (maxhkw@gmail.com)

Harvard University, Massachusetts Institute of Technology, & Diffeo

November 10, 2020

Abstract

Collaboration requires agents to coordinate their behavior on the fly, sometimes cooperating to solve a single task together and other times dividing it up into sub-tasks to work on in parallel. Underlying the human ability to collaborate is theory-of-mind, the ability to infer the hidden mental states that drive others to act. Here, we develop Bayesian Delegation, a decentralized multi-agent learning mechanism with these abilities. Bayesian Delegation enables agents to rapidly infer the hidden intentions of others by inverse planning. We test Bayesian Delegation in a suite of multi-agent Markov decision processes inspired by cooking problems. On these tasks, agents with Bayesian Delegation coordinate both their high-level plans (e.g., what sub-task they should work on) and their low-level actions (e.g., avoiding getting in each other's way). When matched with partners that act using the same algorithm, Bayesian Delegation outperforms alternatives. Bayesian Delegation is also a capable ad-hoc collaborator and successfully coordinates with other agent types even in the absence of prior experience. Finally, in a behavioral experiment, we show that Bayesian Delegation makes inferences similar to human observers about the intent of others. Together, these results argue for the centrality of theory-of-mind for successful decentralized multi-agent collaboration.

Keywords: coordination; social learning; inverse planning; Bayesian inference, multi-agent reinforcement learning

Too many cooks: Bayesian inference for coordinating multi-agent collaboration

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 behaviors. In the absence of prior experience, social roles, and norms, we still find ways to negotiate our joint behavior in any given moment to work together with efficiency (Tomasello, Carpenter, Call, Behne, & Moll, 2005; Misyak, Melkonyan, Zeitoun, & 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. Coordination unfolds over many timescales and these commonsense abilities are at the core of human social intelligence. In order to build social machines we must engineer AI systems that can coordinate with us and with each other as rapidly and as flexibly as people do (Lake, Ullman, Tenenbaum, & Gershman, 2017).

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 et al., 2005; Shum, Kleiman-Weiner, Littman, & Tenenbaum, 2019). We aim to build agents that have these kinds of abilities and show that they are powerful building blocks for coordinated cooperation.

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 & Kraus, 1996; Cohen & Levesque, 1991; Tambe, 1997). Most recently, environments inspired by *Overcooked* have been used in deep reinforcement learning studies where agents are trained by self-play or by imitating human behavior (Song, Wang, Lukasiewicz, Xu, & Xu, 2019; Carroll et al., 2019). In contrast, our approach is based on techniques that dynamically learn during a single interaction 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 & Boutilier, 2003; Stone, Kaminka, Kraus, & Rosenschein, 2010; Barrett, Stone, & Kraus, 2011). Other 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, & Tenenbaum, 2016). Specifically, we build on recent algorithmic progress in Bayesian theory-of-mind (Ramırez & Geffner, 2011; Nakahashi, Baker, & Tenenbaum, 2016; Baker, Jara-Ettinger, Saxe, & Tenenbaum, 2017; Shum et al., 2019) and learning statistical models of others (Barrett, Stone, Kraus, & Rosenfeld, 2012; Melo & 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, & How, 2019). In contrast to models which have explicit communication mechanism or centralized controllers (McIntire, Nunes, & Gini, 2016; Brunet, Choi, & 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 & Durfee, 2004, 2005), but these methods have also been centralized. We draw more closely from decentralized multi-agent planning approaches for best response (Claes et al., 2015; Claes, Oliehoek, Baier, & Tuyls, 2017) and multi-agent plan recognition for spatial inference (Sukthankar & Sycara, 2006; Saria & Mahadevan, 2004). These prior works focus on simpler tasks with spatial sub-tasks called *Spatial Task Allocation Problems* (SPATAPs). There are no mechanisms for agents to cooperate on the same sub-task as each sub-task is spatially distinct.

Here, we develop *Bayesian Delegation*, a new 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 three player teams in a constrained space. Second, we show Bayesian Delegation is a powerful ad-hoc collaborator. It performs better than alternatives when paired with agents of a different type. Finally, in a behavioral experiment, human participants observed others interact and made inferences about the sub-tasks they were working on. Bayesian Delegation aligned with many of the fine-grained variations in human judgments. Although the model was never trained on human data or other agents' behavior, it was the best ad-hoc collaborator and predictor of human inferences.

Multi-Agent MDPs with Sub-Tasks

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

Unlike traditional multi-agent Markov decision process, the environments we study have a partially ordered set of sub-tasks $\mathcal{T} = \{\mathcal{T}_0 \ldots \mathcal{T}_{|\mathcal{T}|}\}$ which is generated by representing each recipe as an instance of the planning language STRIPS (Fikes & Nilsson, 1971). Each instance consists of an initial state, a specification of the goal state, and a set of sub-tasks $\mathcal{T}_i$. Each sub-task has preconditions that specify when the 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. Unlike in SPATAPs, both `X` and `Y` can be movable. In the cooking environments we study here, the partial order of sub-tasks refers to a "recipe". Figure 1 shows an example of sub-task partial orders for a recipe. Our task structures are inspired by prior work on both the cooking domain (Grosz & Kraus, 1996; Carroll et al., 2019;

Song et al., 2019) and others (Amato et al., 2019; Guestrin, Venkataraman, & Koller, 2002).

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 sub-tasks are the `Merge` operators in Figure A1a. A plan for a STRIPS instance is a sequence of high-level actions that can be executed from the initial state and results in a goal state. Examples of such plans are shown in Figure A1. 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 explore all shortest "recipe paths" between the two states. The environment then returns $\mathcal{T}$, the set of possible high-level actions which terminate at a completed 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 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 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 to effectively complete the task.

The Overcooked Coordination Test Suite

We now describe the Overcooked-inspired environments we use as a test suite for evaluating multi-agent collaboration[†]. The goal in each environment is to cook a recipe in as few time steps as possible. The episode terminates after either the agents bring the finished recipe dish to the star square or 100 time steps elapse. Each environment is a 2D grid-world kitchen. Figure 1a shows an

[†] All code for the environments and select videos of the agents can be found at `https://github.com/rosewang2008/gym-cooking`.

example kitchen layout and Figure A1 shows the full set of recipes used in our evaluations.

The kitchens are built from counters that contain both movable food and plates and immovable stations (e.g., knife stations). The state is represented as a list of entities and their type, location, and status (Diuk et al., 2008). See Table A1 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 objects to each other.

This test suite allows us to evaluate models based on the coordination challenges raised in the introduction. The recipes assess rapid convergence—for instance, the *Salad* recipe can be assembled in multiple ways and agents' plans must align—and the spatial layouts provide opportunities for multiple agents to work together advantageously and/or avoid navigational obstacles. Thus, these environments enable us to study multi-agent coordination across levels of hierarchical planning.

Computational Model: 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 on each sub-task, and an action planner that finds approximately optimal policies that implement divide-and-conquer or cooperation. 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. Probabilistic inference of the sub-tasks others are working on enables each agent to select the best 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 allocations of each agent to a sub-task. Let **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 $\mathbf{ta} = [(i : \mathcal{T}_2, j : \mathcal{T}_2), (i : \mathcal{T}_2, j : \mathcal{T}_1), (i : \mathcal{T}_1, j : \mathcal{T}_1), (i : \mathcal{T}_2, j : \mathcal{T}_2)]$ where $i : \mathcal{T}_1$ means that agent i is "delegated" to sub-task $\mathcal{T}_1$. Thus, **ta** includes both the possibility that agents will divide and conquer (work on separate sub-tasks) and the possibility that they will 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, so they maintain 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 model-based reinforcement learning (described below). The posterior is computed by Bayes rule:

$$P(ta|H_{0:T}) \propto P(ta)P(H_{0:T}|ta) = P(ta)\prod_{t=0}^{T} P(\mathbf{a_t}|s_t, ta) \tag{1}$$

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 (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.

$$P(\mathbf{a_t}|s_t, ta) \propto \prod_{i:\mathcal{T} \in ta} exp(\beta * Q^*_{\mathcal{T}_i}(s, a_i)) \tag{2}$$

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 of reward accounts for non-optimal and variable behavior as is typical in other

Bayesian theory-of-mind work (Kleiman-Weiner et al., 2016; Baker et al., 2017; Shum et al., 2019). β controls the degree to which an agent believes others are perfectly optimal. When $\beta \rightarrow 0$, the agent believes others are acting randomly. When $\beta \rightarrow \infty$, the agent believes others are perfectly maximizing. Since the likelihood is computed by a model based planner, 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 each agent's observation of 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 returns ta that can be accomplished in less time a higher prior probability. Priors are reinitialized when new sub-tasks have their preconditions satisfied and when others are completed.

Action planning transforms sub-task allocations into efficient actions and provides the critical likelihood for Bayesian Delegation (see Equation 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, & 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 a cost of 1 and movement (as opposed to staying still) by an additional cost of 0.1. This cost structure incentivizes efficiency in movement. The lower-bound was initialized to the Manhattan distance between objects (which ignores barriers). The upper-bound was initialized to 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 sub-task 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 update V and Q, see McMahan et al. (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 $\beta = 1.3$ (this value was chosen to be consistent with a behavioral experiment described below). At each time step, agents select the action with the highest value for their sub-task. When agents have no valid sub-tasks, they take a random action (uniform across the four directions of movement and stay-in-place actions). This random motion improves the performance of the alternative models since the agents often get stuck and block each other from completing the recipe. The random motion does not impact Bayesian Delegation because agents are always assigned to a sub-task.

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 & Leyton-Brown, 2010; Kleiman-Weiner et al., 2016; Shum et al., 2019).

When $\mathcal{T}_i = \mathcal{T}_{-i}$, agent i attempts to work together on the same sub-task with the other agent(s). To do this, the agent simulates a fictitious centralized planner that controls the actions of all agents working together on the same sub-task (Kleiman-Weiner et al., 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 planning stochasticity. Note that although we use BRTDP, any other model-based reinforcement learner or planner could be used in its place.

Results

We evaluate the performance of Bayesian Delegation across three different experimental paradigms. First, we test the 2-agent and 3-agent performance of each method paired with itself (homogeneous teams). Second, we test the "ad-hoc" performance of each method when paired with an agent of a different method (heterogeneous teams). Finally, we test each model's ability to predict human inferences of sub-task allocation after observing the behavior of other agents (human inferences).

We compare the performance of Bayesian Delegation (BD) to four alternative baseline agents: Uniform Priors (UP), which puts a 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 & 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 et al., 2015) because agents cannot share sub-tasks and D&C discounts sub-tasks most likely to be attended to by other agents proportional to $P(ta|H)$); Greedy, which selects the sub-task it can complete most quickly without considering the sub-tasks other agents are working on. End-to-end optimization of the full recipe using techniques such as DQN (Mnih et al., 2013) and Q-learning (Watkins & 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 **ta**, but places uniform probability across all elements, i.e., $P(ta) = \frac{1}{4}$ for all $ta \in \mathbf{ta}$. FB proposes the same **ta** with the same priors as Bayesian Delegation, but never updates its beliefs. The D&C agent does not allow for joint sub-tasks, so it proposes $\mathbf{ta} = [(i : \mathcal{T}_1, j : \mathcal{T}_2), (i : \mathcal{T}_2, j : \mathcal{T}_1)]$. Lastly, Greedy makes no inferences; each agent i reduces the set to $\mathbf{ta} = [(i : \mathcal{T}_1), (i : \mathcal{T}_2)]$. 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. Agents are evaluated in 9 different kitchen-recipe combinations (3 recipes $\times$ 3 kitchens).

Homogeneous teams

Table 1 quantifies the performance of all agents (2- and 3-agent teams) aggregated across the 9 environments. Bayesian Delegation outperforms all alternative models and completes recipes with less time steps and fewer shuffles. The performance gap between Bayesian Delegation and the alternative models 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 2 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 outperforms 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. Alternative agents were also less capable of low-level coordination, resulting in more shuffles.

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 either be combined together and then plated, the lettuce can be plated first and then the tomato added, or the tomato can be plated first and then the lettuce added. The second challenge is that no agent can perform all the sub-tasks alone, thus they must converge to the same order. Unless the non-learning agents coordinate by luck, they will not recover. 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 Delegation can break these symmetries by yielding to others so long as the group makes 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.

Heterogeneous teams

Next, we evaluated the ad-hoc performance of the agents, where each agent was paired with the other agent types (heterogeneous teams). Our results show that Bayesian Delegation is a competent ad-hoc collaborator. None of the agents had any prior experience with the other agents. Figure 3 shows the performance of each agent when matched with each other type and in aggregate across all recipe-kitchen combinations. When Bayesian Delegation was 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 Bayesian Delegation can learn in-the-moment, it can overcome some of the ways that the alternative agents get stuck. UP performs better when paired with Bayesian Delegation or FB compared to the homogeneous team results, 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 types are all capable of representing cooperative plans where both agents cooperate on the same sub-task.

Human inferences

Finally, we quantitatively compared the inferences made by Bayesian Delegation to those made by people in these same settings. 60 participants were recruited from Amazon Mechanical Turk where they observed scenes (a few time steps) of two agents working together (see Figure 4) and were then asked to make probabilistic judgements about which sub-tasks each agent was carrying out as the interactions unfolded. The stimuli include a variety of coordinated plans such as instances of clear task allocation (e.g., Figure 4a) and of ambiguous plans where the agent intentions become more clear over time as the interaction continues (e.g., Figure 4d). Each participant made 51 distinct judgments. We measured the correlation coefficient (R) between mean participant judgements and beliefs formed by our model ($P(ta|H)$) after observing the same trajectory of interactions H.

In Figure 4, we show the six scenes presented to participants in the experiment, along with the mean of participant judgements and inferences made by Bayesian Delegation at each time step. Generally, the model captures people's relative beliefs about which task allocations are most probable given the interactions observed so far. For some trajectories (e.g. Figures 4a, 4f), the model is much more confident about one particular task allocation than the rest, but in all cases this still corresponds with the most probable allocation inferred by people as well. Even when priors are misaligned (such as in Figure 4d), Bayesian Delegation is able to correctly update its beliefs by the end of the trajectory.

We further compared the predictions made by all five models and quantified the overall correspondence of each model with human inferences. Figure 5 shows that all four alternative models are less aligned with human judgements than Bayesian Delegation is. In particular, priors may help capture people's initial beliefs about the high-level plans agents have, while belief updates may be important for tracking how people's predictions evolve over the course of an interaction. For instance, updates are especially important in Figure 4d where the observed actions are initially ambiguous. When the left agent moves to put the tomato on the counter at time step $t = 2$; this may be interpreted as either passing `Merge(Tomato.chopped, Plate[])` or `Merge(Lettuce.unchopped, Knife)`, but after only a few time steps, the former emerges as more probable. With only a single free parameter (β) optimized for correlation with human data, Bayesian Delegation captures the fine-grained structure of human sub-task inferences. These results suggest Bayesian Delegation may help us build better models of human theory-of-mind, enabling machines to effectively cooperate with people.

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, which allows them to determine when they should help another agent on the same sub-task and when they should work divide-and-conquer for increased efficiency. Bayesian Delegation also enables agents to complete sub-tasks that neither agent could achieve on its own. These features reflect many natural aspects of human theory-of-mind and cooperation (Tomasello, 2014). Indeed, like people, it makes predictions about sub-task allocations from only sparse data, and does so in ways consistent with human judgments.

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 both lettuce and tomato across the divider for the other to chop it. After dropping off the lettuce, the first agent should reason that it has fulfilled its role in that joint plan and can move on to next task, i.e., that the rest of the sub-task depends only on the actions of the other agent. If agents were able to recognize when their own specific roles in sub-tasks are finished they could look ahead to future sub-tasks that will need to be done even before their preconditions are satisfied. At some point, as one scales up the number of agents, there can be “too many cooks” in the kitchen! Other, less flexible mechanism and representations will likely play a crucial role in coordinating the behavior of larger groups of agents such as hierarchies, norms, and conventions (Young, 1993; Bicchieri, 2006; Lewis, 1969; Lerer & Peysakhovich, 2019). These representations are essential for building agents that can form longer term collaborations which persist beyond a single short interaction and are capable of partnering with human teams and with each other.

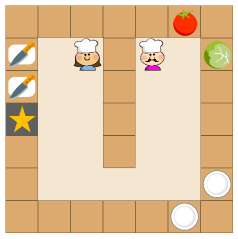
Acknowledgements

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).

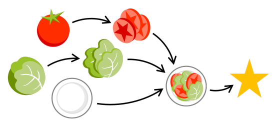
Table 1

Performance of Bayesian Delegation and alternative models with two and three agents when all agents use the same algorithm (homogeneous teams) . All metrics are described in the text. See Figure 2 for detailed results of Time Steps and Completion. Averages ± standard error of the mean (N = 20 *random seeds each).*

		Time Steps (↓ better)	Completion (↑ better)	Shuffles (↓ 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) *Partial-Divider*.



Goal

`Delivery[Plate[Tomato.chopped, Lettuce.chopped]]`

(b) *Salad* recipe.

```
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)
```

(c) Example sub-task order for *Salad*.

***Figure 1*.** The Overcooked environment. (a) The *Partial-Divider* kitchen offers many counters for objects, but forces agents to either move through a narrow bottleneck or pass objects across the divider. (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 orders that each part of the recipe can be completed. All sub-tasks are expressed using the `Merge` operator. By combining different recipes with different kitchens, we can generate combinatorial variation in high-level goals and low-level navigation challenges.

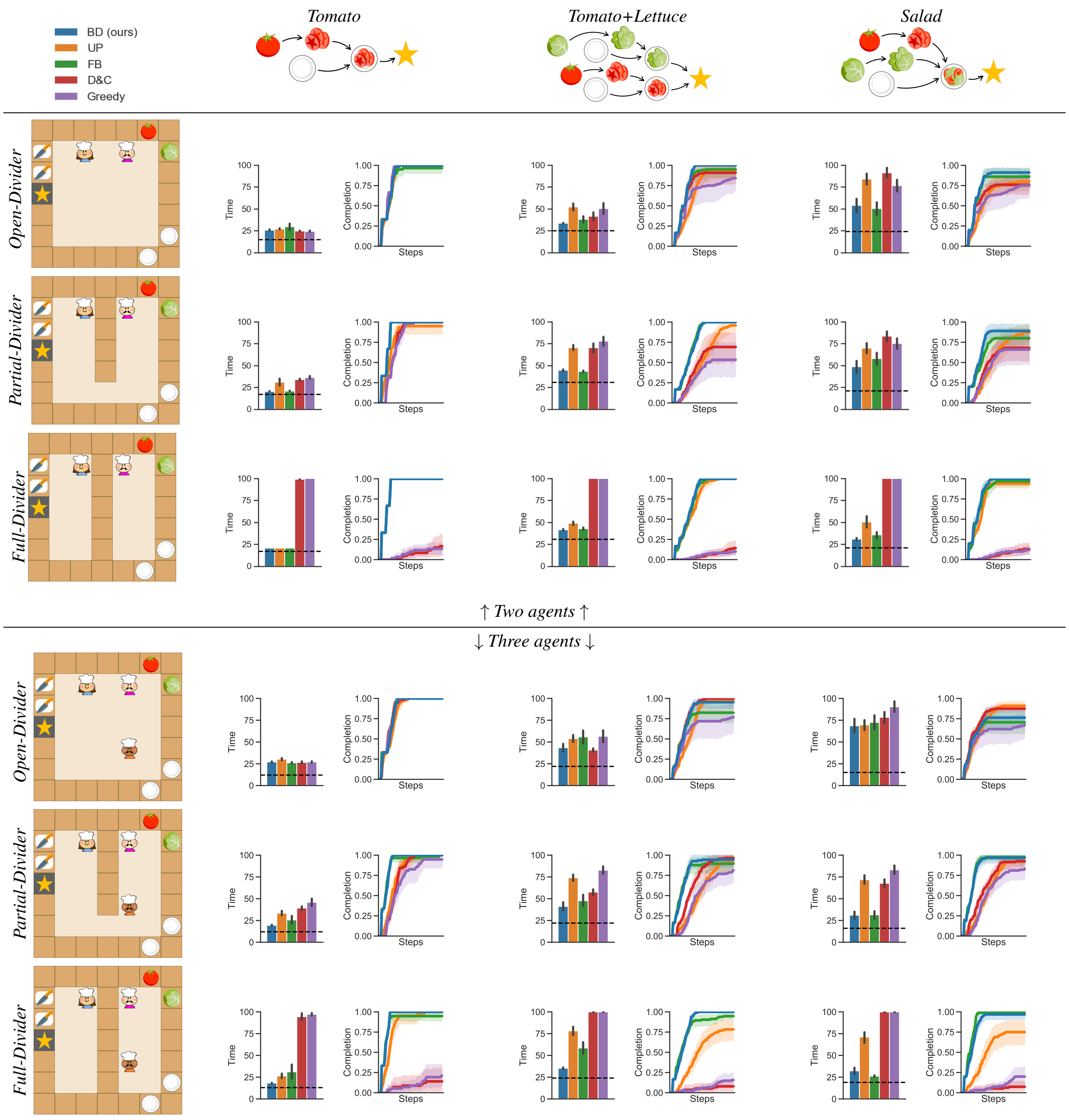
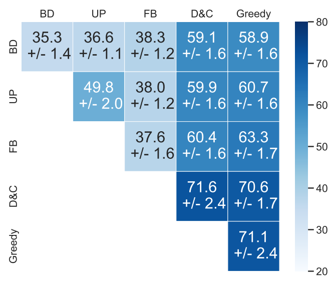


Figure 2. Performance of each agent type results for each kitchen-recipe (lower times and faster completions are better) for homogeneous teams of two and three agents. The row shows the kitchen and the column shows the recipe. Within each kitchen-recipe (row-column), the left graph shows the number of time steps needed to complete all sub-tasks. The dashed lines on the left graph represent the optimal performance of a fully-centralized team. The right graph shows the fraction of sub-tasks completed over time. The Bayesian Delegation agent completes more sub-tasks and does so more quickly compared to the alternatives ($N = 20$ random seeds each).



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

Figure 3. Ad-hoc performance of different. (Left) Rows and columns correspond to different agent types. Each cell is the average number of time steps (the lower/lighter the better) of the row agent type teamed with the column agent type. Results are averaged across all nine 2-agent kitchen-recipes ($N = 20$ random seeds each). (Right) Average performance ($\pm$ standard error of the mean) of agents when paired with the others.

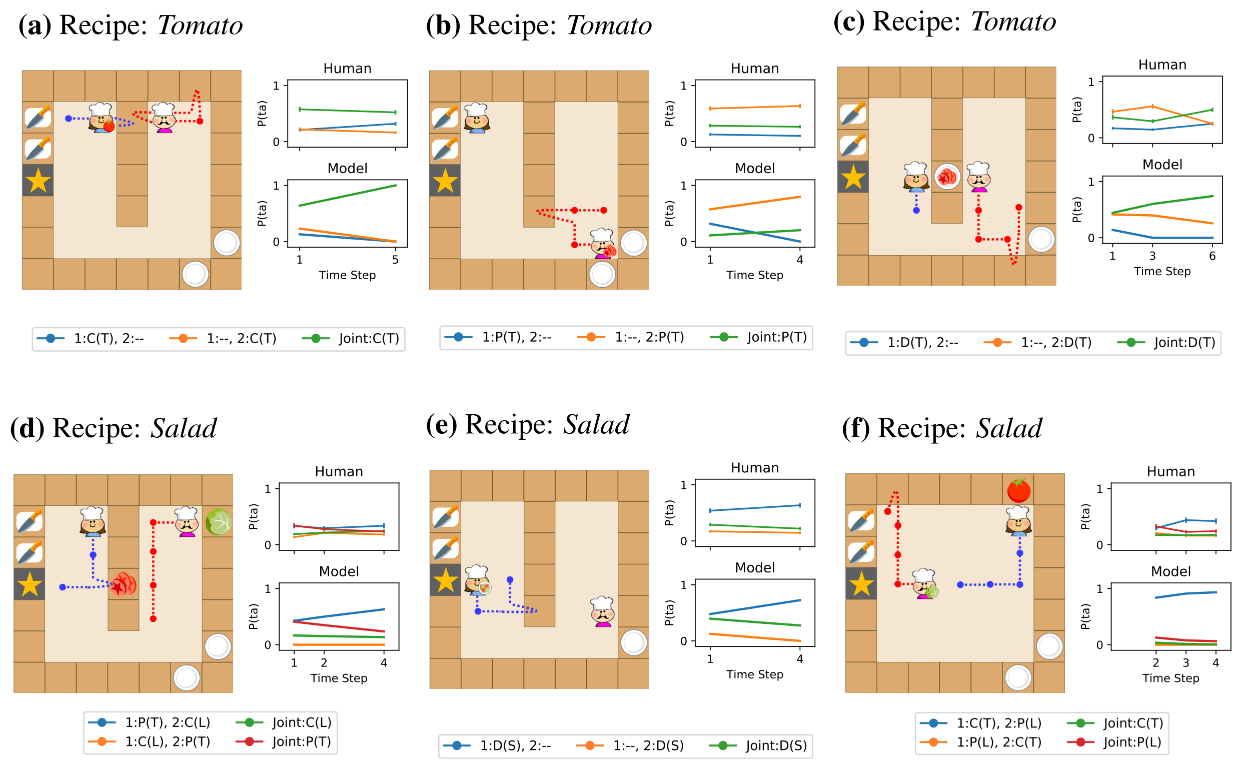


Figure 4. Scenes and results for the human inferences experiment. Each agent's past trajectory is illustrated by a dotted path, with sharp curves into counters representing picking up or putting down an object. To the right of each trajectory are the inferences made by the model (Bayesian Delegation) and the average participant judgment. The legend notes the possible task allocations of agents (1 or 2) working individually or together (Joint): C = chop, P = plate, D = deliver, T = tomato, L = lettuce, and S = salad. E.g., 1:C(T) refers to Agent 1 chopping the tomato. Error bars are the standard error of the mean.

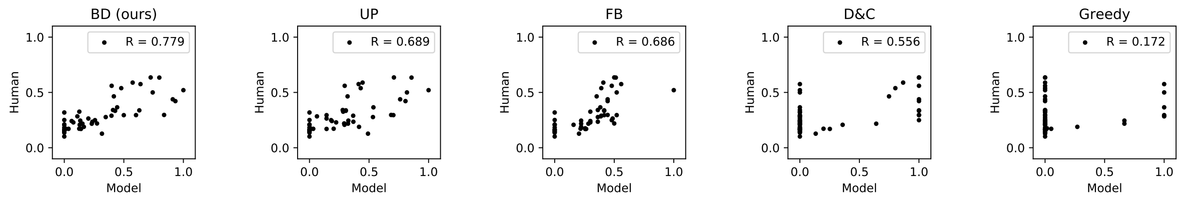


Figure 5. Correlation between model and human inferences. Our model (BD, far left) is more aligned with human judgements than the alternative models.

Appendix

Additional Figures

References

Amato, C., Konidaris, G., Kaelbling, L. P., & How, J. P. (2019). Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research*, *64*, 817–859.

Baker, C. L., Jara-Ettinger, J., Saxe, R., & Tenenbaum, J. B. (2017). Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, *1*, 0064.

Barrett, S., Stone, P., & Kraus, S. (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).

Barrett, S., Stone, P., Kraus, S., & Rosenfeld, A. (2012). Learning teammate models for ad hoc teamwork. In *Aamas adaptive learning agents (ala) workshop* (pp. 57–63).

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

Boutilier, C. (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).

Brunet, L., Choi, H.-L., & How, J. (2008). Consensus-based auction approaches for decentralized task assignment. In *Aiaa guidance, navigation and control conference and exhibit* (p. 6839).

Carroll, M., Shah, R., Ho, M., Griffiths, T., Seshia, S., Abbeel, P., & Dragan, A. (2019). On the utility of learning about humans for human-ai coordination. In *Advances in neural information processing systems.*

Chalkiadakis, G., & Boutilier, C. (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, D., Oliehoek, F., Baier, H., & Tuyls, K. (2017). Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the 16th conference on* *autonomous agents and multiagent systems* (pp. 492–500).

Claes, D., Robbel, P., Oliehoek, F. A., Tuyls, K., Hennes, D., & Van der Hoek, W. (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).

Cohen, P. R., & Levesque, H. J. (1991). Teamwork. *Noûs*, *25*(4), 487–512.

Cox, J. S., & Durfee, E. 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).

Cox, J. S., & Durfee, E. 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, C., Cohen, A., & Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on machine learning* (pp. 240–247).

Ephrati, E., & Rosenschein, J. S. (1994). Divide and conquer in multi-agent planning. In *Aaai* (Vol. 1, p. 80).

Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(3), 189 - 208. doi: https://doi.org/10.1016/0004-3702(71)90010-5

Ghost Town Games. (2016). *Overcooked.*

Grosz, B. J., & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, *86*(2), 269–357.

Guestrin, C., Venkataraman, S., & Koller, D. (2002). Context-specific multiagent coordination and planning with factored mdps. In *Aaai/iaai* (pp. 253–259).

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

Kleiman-Weiner, M., Ho, M. K., Austerweil, J. L., Littman, M. L., & Tenenbaum, J. 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.*

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*.

Lerer, A., & Peysakhovich, A. (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).

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

McIntire, M., Nunes, E., & Gini, M. (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. B., Likhachev, M., & Gordon, G. 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).

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

Misyak, J. B., Melkonyan, T., Zeitoun, H., & Chater, N. (2014). Unwritten rules: virtual bargaining underpins social interaction, culture, and society. *Trends in cognitive sciences*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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

Nakahashi, R., Baker, C. L., & Tenenbaum, J. B. (2016). Modeling human understanding of complex intentional action with a bayesian nonparametric subgoal model. In *Aaai* (pp. 3754–3760).

Ramırez, M., & Geffner, H. (2011). Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *Ijcai* (pp. 2009–2014).

Saria, S., & Mahadevan, S. (2004). Probabilistic plan recognition in multiagent systems. In *Icaps* (pp. 287–296).

Shum, M., Kleiman-Weiner, M., Littman, M. L., & Tenenbaum, J. 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, Y., Wang, J., Lukasiewicz, T., Xu, Z., & Xu, M. (2019). Diversity-driven extensible hierarchical reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 4992–4999).

Stone, P., Kaminka, G. A., Kraus, S., & Rosenschein, J. S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-fourth aaai conference on artificial intelligence.*

Sukthankar, G., & Sycara, K. (2006). Simultaneous team assignment and behavior recognition from spatio-temporal agent traces. In *Aaai* (Vol. 6, pp. 716–721).

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

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

Tomasello, M., Carpenter, M., Call, J., Behne, T., & Moll, H. (2005). Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and brain sciences*, *28*(05), 675–691.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Wright, J. R., & Leyton-Brown, K. (2010). Beyond equilibrium: Predicting human behavior in normal-form games. In *Aaai.*

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

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 $\rightarrow$ Food.chopped + Knife

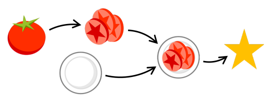
Food1 + Food2 $\rightarrow$ [Food1, Food2]

X + Y[] $\rightarrow$ Y[X]

Table A1

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.*

(a)



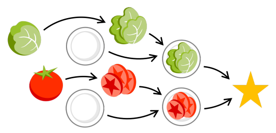
Goal

`Delivery[Plate[Tomato.chopped]]`

Example Plan

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

(b)



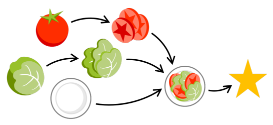
Goal

`Delivery[Plate[Tomato.chopped],`
`Plate[Lettuce.chopped]]`

Example Plan

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

(c)



Goal

`Delivery[Plate[Tomato.chopped,`
`Lettuce.chopped]]`

Example Plan

```
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)
```

Figure A1. Recipes and example partial orderings. All sub-tasks are expressed in the `Merge` operator. In (a) *Tomato*, the task is to take an unchopped tomato and then chop, plate, and deliver it. In (b) *Tomato+Lettuce*, the task builds on *Tomato* and adds chopping, plating, and delivering a piece of lettuce. In (c) *Salad*, the two chopped foods are combined on a single plate and delivered. The example plans show one of many possible orderings for completing the recipe.