Preprint accepted at the
Thirtieth AAAI Conference on
Artificial Intelligence (AAAI-2016),
Phoenix, Arizona, 2016.

ConTaCT : Deciding to Communicate during Time-Critical Collaborative Tasks in Unknown, Deterministic Domains

Vaibhav V. Unhelkar and **Julie A. Shah**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
{unhelkar, julie_a_shah}@csail.mit.edu

Abstract

Communication between agents has the potential to improve team performance of collaborative tasks. However, communication is not free in most domains, requiring agents to reason about the costs and benefits of sharing information. In this work, we develop an online, decentralized communication policy, ConTaCT, that enables agents to decide whether or not to communicate during time-critical collaborative tasks in unknown, deterministic environments. Our approach is motivated by real-world applications, including the coordination of disaster response and search and rescue teams. These settings motivate a model structure that explicitly represents the world model as initially unknown but deterministic in nature, and that de-emphasizes uncertainty about action outcomes. Simulated experiments are conducted in which ConTaCT is compared to other multi-agent communication policies, and results indicate that ConTaCT achieves comparable task performance while substantially reducing communication overhead.

Introduction

Communication between agents has the potential to improve team performance during collaborative tasks, but often has associated costs. These costs may arise due to the power requirements necessary to transmit data, computational requirements associated with processing new data, or the limitations of human information processing resources (if the team in question includes human agents). The benefit gleaned from using newly communicated information may not necessarily outweigh the associated costs, and excessive communication can hamper collaborative task performance.

A number of works (Xuan, Lesser, and Zilberstein 2004; Spaan, Gordon, and Vlassis 2006; Williamson, Gerding, and Jennings 2009) have aimed to design communication strategies that support agents in communicating only when necessary, reducing communication overhead and potentially improving collaborative task performance. Prior decision-theoretic approaches for generating online communication (Roth, Simmons, and Veloso 2005; Wu, Zilberstein, and Chen 2011) have largely focused on tasks modeled using extensions of DEC-POMDP (Bernstein et al. 2002) that include communications (Pynadath and Tambe 2002; Goldman and Zilberstein 2003) and assume complete knowledge of action and sensing uncertainty present in the model. These approaches are particularly suited for multi-agent settings that include uncertain outcomes from agents' actions and partially observable local states, and circumstances where the associated uncertainty - namely, transition and observation probabilities - can be quantified a priori.

The focus of our work, in contrast, is a subclass of decentralized, multi-robot problems wherein the world model is initially unknown but deterministic in nature. In the DEC-POMDP model, this corresponds to a setting with a deterministic but initially unknown transition function. We are motivated by real-world applications, including the coordination of disaster response teams, where agents can often achieve the desired outcome from a chosen action in a robust fashion (e.g., through the use of dynamic controllers), thereby de-emphasizing uncertainty about the outcomes of actions. However, the world model is unknown at the outset, potentially precluding a priori generation of optimal or even feasible plans. In addition, these domains are typically time-critical, meaning that there are hard temporal deadlines under which tasks must be completed. Finally, agents are assumed to have prior knowledge of the planning behavior, initial states and goal states of their collaborators.

The key contribution of this paper is an online, decentralized communication strategy, ConTaCT, that allows an agent to reason about whether to communicate to team members during execution of a time-critical collaborative task within an unknown, deterministic domain. By maintaining an estimate of what other team members know about the domain, the algorithm allows the agent to compare the expected benefit of its decision against the cost of communication.

We begin with a formal definition of the subclass of the multi-agent problems of interest, and highlight how it differs from problems considered in prior art. Next, we describe the ConTaCT algorithm and evaluate its performance of a simulated task motivated by rescue operations in disaster response. We compare ConTaCT with online communication policies generated using an existing approach for multi-agent communication (Roth, Simmons, and Veloso 2005) applied to our task model, and show that ConTaCT achieves comparable task performance while substantially reducing communication overhead.

Problem Definition

In this section, we define the decentralized, multi-agent communication problem in the context of time-critical collaborative tasks within unknown, deterministic domains. We specify the model structure, objective function and solution representation. In the following section, we highlight the need to study the chosen subclass of communication problems and call attention to its key differences from those assessed in prior research.

Model Structure

Our model builds upon the factored, DEC-MDP framework (Becker et al. 2004), which is transition and observation independent, locally fully observable and has null state space corresponding to external features (Becker et al. 2004). We incorporate additional model features to represent communication between agents and characteristics of time-critical tasks. The model, which we call the "time-critical deterministic factored, DEC-MDP with communication" (TCD-DEC-MDP-COM) model, is defined as follows:

- I is a finite set of agents i indexed $1, \ldots, n$.
- $T \in \mathbb{Z}^+$ denotes the time horizon of the problem.
- S_i is a finite set of states, s_i, available to agent i. $\vec{S} = \times_{i \in I} S_i$ is the set of joint states, where $\vec{s} = \langle s_1, \ldots, s_n \rangle$ denotes a joint state.
- $a_i \in A_i$ is a finite set of operation actions, a_i, available to agent i. $\vec{A} = \times_{i \in I} A_i$ is the set of joint actions, where $\vec{a} = \langle a_1, \ldots, a_n \rangle$ denotes a joint action.
- P_i is a Markovian state transition function ($P_i : S_i \times A_i \times S_i \rightarrow \{0, 1\}$), corresponding to agent i, which depends on local states and actions. The joint state transition function is given as $P = \prod_{i \in I} P_i$, as we assume transition independence. Note that P_i maps only to the discrete set $\{0, 1\}$, denoting deterministic transitions.
- $C_i \in \{0, 1\}$ is the set of communication decisions available to the agent i. $\vec{C} = \times_{i \in I} C_i$ is the set of joint communication decisions. The communication decision 0 represents no communication, while communication decision 1 indicates transmission of information to all agents. In order to model the communication delays observed in practice, we assume that communication is not instantaneous and that information takes up to one time step to reach other agents. At a given time step an agent may carry out an operation action and communication simultaneously.
- R_{ia} is the agent action reward function ($R_{ia} : S_i \times A_i \rightarrow \Re$), corresponding to agent i. It denotes the reward received by agent i while initializing from state $s_i(t)$ and taking action $a_i(t)$.
- R_{ic} is the agent communication reward function ($R_{ic} : S_i \times C_i \rightarrow \Re$), corresponding to agent i. It denotes the reward received by agent i for making communication decision $c_i(t)$, while in state $s_i(t)$.
- R_{ig} is the agent goal reward function ($R_{ig} : S_i \rightarrow \Re$) corresponding to agent i. It denotes the reward received by agent i based on its terminal state $s_i(T)$. The agent goal reward function is used to model the time-critical nature of the task, and can serve to quantify the penalty for not completing the subtask by the specified time.
- Ξ_i is the cumulative agent reward corresponding to agent i, and is given as

$$\Xi_i = R_{ig}\left(s_i(T)\right) + \sum_{t=0}^{T} R_{ia}\left(s_i(t), a_i(t)\right) + R_{ic}\left(c_i(t)\right)$$

- Ξ is the final team reward assigned at the end of the task, and is given as $\Xi = \min_{i \in I} \Xi_i$. This definition of team reward is chosen to reflect that the overall task is complete if and only if all agent subtasks are complete.

Task Characteristics

Our model also incorporates following task characteristics[1],

- The agents begin the task with incomplete information about the domain. The true deterministic, Markovian state transition function P is not accurately known to the agents at the start of the problem, but an initial estimate of the state transition function $\hat{P}$ is common knowledge.
- On visiting a state s, an agent observes its own state as well as some components of the transition function based on a domain-dependent function ObserveDomain. The ObserveDomain function depends on the previous estimate of transition function $\hat{P}^i(t-1)$ and the local state of the agent $s_i(t)$, and provides an updated estimate $\hat{P}^i(t)$.
- The initial state $\vec{s}(0)$ and desired goal state $\vec{s}_g(T)$ for all agents are pre-specified and common knowledge. The time horizon, action and communication space, reward functions and ObserveDomain function are also known to the agents.
- The algorithm to generate action plans (i.e., the planning technique) for each agent is specified and common knowledge, and is denoted as ActionPlanner. This planner completely specifies the policy $\pi_i : S_i \rightarrow A_i$ of an agent i given the initial state $s_i(0)$, goal state $s_{g,i}(T)$ and transition function $\hat{P}^i$ used to generate the plan.
- Lastly, a communication message from an agent i includes the agent's local estimate of transition function $\hat{P}^i$, an indicator function Δ_i corresponding with the observed components of P that it knows accurately, the joint state $\hat{\vec{s}}^{\,i}$ and its action policy π_i.

Objective Function The objective of the agents is to complete the task with maximum reward, represented as follows:

$$\max \Xi = \max \min_{i \in I} \Xi_i.$$

Solution Representation

Given the model structure, task characteristics and objective function, our objective is to design a decentralized algorithm that determines at each time step whether an agent should re-plan for itself using newly observed information about the environment and/or communicate this information to others, in order to maximize team reward.

[1]Notation: $\hat{\cdot}$ denotes an estimate. A superscript denotes the agent maintaining the estimate. Two superscripts are equivalent to nested superscripts. For instance, $\hat{P}^{ji} = (\hat{P}^j)^i$ and denotes estimate maintained by agent i regarding the estimate of P maintained by agent j.

Background and Prior Art

We review prior art related to multi-agent communication, including task models and communication algorithms, and discuss the distinguishing features of our approach.

Task Models The first decision-theoretic frameworks to describe multi-agent tasks with communication include COM-MTDP (Pynadath and Tambe 2002) and DEC-POMDP-COM (Goldman and Zilberstein 2003). These approaches extended the DEC-POMDP (Bernstein et al. 2002) model to include communication actions and their associated costs.

These models have since been augmented in a number of ways. For example, the work of Spaan, Gordon, and Vlassis modeled the noise within the communication channel. The DEC-POMDP-Valued-COM framework (Williamson, Gerding, and Jennings 2008) augmented the model to include communication rewards corresponding to information gain, and used a heuristic to determine the contribution of that communication to the total team reward. RS-DEC-POMDP improved upon this by adopting a more principled approach to merge communication and action rewards (Williamson, Gerding, and Jennings 2009). Most recently, Amato, Konidaris, and Kaelbling developed a framework incorporating macro actions for DEC-POMDPs capable of generating both actions and communication (Amato et al. 2014).

TCD-DEC-MDP-COM, the task model proposed in this paper, addresses a sub-class of multi-agent communication problems by building upon the factored DEC-MDP model (Becker et al. 2004) and incorporating communication actions and costs similar to DEC-POMDP-COM. Our model is distinguished by additional features that represent time-critical tasks and unknown but deterministic settings. By considering deterministic, factored DEC-MDP tasks, our model does not include sensing or action uncertainties, thereby improving computational tractability. However, from the agent's perspective, we treat the transition function as an unknown. Our model includes an additional terminal reward function to model the associated time constraints; an alternate but potentially computationally prohibitive approach would be to include time-dependent reward functions by incorporating time as part of the state. Also, our model assumes non-instantaneous, one-way communication to better capture the impact of communication delays observed in practice.

Communication Algorithms Several prior approaches have used sequential decision theory to determine when agents should communicate in multi-agent settings. These methods have ranged from offline, centralized approaches that generate communication policies prior to task initiation (Nair, Roth, and Yokoo 2004; Mostafa and Lesser 2009) to decentralized algorithms that determine each agent's communications during task execution (Xuan, Lesser, and Zilberstein 2004). Here, we review the online, decentralized approaches, due to our focus on finite-horizon, time-critical tasks for which agents must generate plans and make decisions about communication during execution.

Roth, Simmons, and Veloso developed the DEC-COMM algorithm, one of the first approaches to consider communication decisions during execution time. The DEC-COMM algorithm requires agents to generate a centralized, offline policy prior to execution. The agents maintain a joint belief over the state of every agent and execute actions based on the offline policy. Upon making observations about the environment, the agents weigh the expected benefit of sharing this information against the cost of communication to decide whether or not to communicate. Communication messages include the agent's observation history, and are assumed to be transmitted instantaneously. This may result in multiple communications at each time step. Once the information is shared, agents re-compute the joint policy and follow it until the next communication action. By using a pre-computed joint policy and by not using local information until it is communicated, agents maintain perfect predictability and coordination. (Roth, Simmons, and Veloso 2005) also presented a variant DEC-COMM-PF that uses a particle filter for maintaining estimates of possible joint beliefs to improve computational tractability.

Wu, Zilberstein, and Chen designed MAOP-COMM, an online communication algorithm that offers improved computational tractability and performance. The algorithm requires that agents use only jointly known information when generating plans; however, agents are able to use local information for task execution. The proposed communication policy requires agents to communicate when their observations of the environment are inconsistent with their pre-existing beliefs. The MAOP-COMM algorithm is robust to communication delays; however, it assumes that communication occurs according to the synced communication model (Xuan, Lesser, and Zilberstein 2004). Recently, an alternate approach to communication has been developed by posing it as a single agent in a team decision (SATD) problem (Amir, Grosz, and Stern 2014). The authors proposed MDP-PRT, a novel logical-decision theoretic approach that combines MDPs and probabilistic recipe trees (Kamar, Gal, and Grosz 2009).

The decision-theoretic approaches to communication discussed above assume knowledge of the underlying transition and observation probabilities, which is often not available in real-world settings. Our problem definition assumes deterministic transitions and perfect sensing on the part of each agent, and instead requires agents to generate communication decisions in the absence of complete knowledge of the deterministic transition function.

Communication for Time-Critical Collaborative Tasks : ConTaCT

In this section, we describe the proposed algorithm, ConTaCT, for the multi-agent problem of interest - namely, TCD-DEC-MDP-COM with unknown transition function. The algorithm operates in a decentralized fashion and allows each agent on a team to make communication and re-planning decisions during task execution, with the aim of maximizing the team reward. The proposed algorithm includes the following three components:

- the model representation maintained by each agent;
- algorithms to update the model with and without communication (model 'propagate' and 'update' rules); and
- an algorithm to generate communication and trigger re-planning when warranted.

Essentially, the ConTaCT algorithm computes, at each time step, the anticipated reward for communicating and re-planning by maintaining an estimation of the knowledge of the other agents. The output of the algorithm is the decision whether or not to communicate and/or re-plan at each time step. Below, we describe each of these three components.

Agent Model Representation

In order to carry out the multi-agent collaborative task, each agent i must maintain its local action policy π_i - also referred to as the agent's plan. In addition to this, the ConTaCT algorithm requires the agent i to maintain estimates of the following:

- $\hat{\vec{s}}^i$, the joint state, and $\hat{P}^i$, the transition function;
- $\hat{\pi}^i_j$, the action policy of other agents $(j \in I \setminus i)$;
- $\hat{P}^{ji}$, the transition function as estimated by other agents j;
- Δ_i, an indicator function $(\Delta_i : S \times A \times S \rightarrow \{0, 1\})$ defined over the same space as P, which maps to 1 if the corresponding component of transition function is known to i accurately and 0 otherwise; and
- Δ^i_j indicator function of other agents $(j \in I \setminus i)$.

We denote the agent model as $M_i : \langle \hat{\vec{s}}^i, \hat{P}^i, \hat{P}^{ji}, \pi_i, \hat{\pi}^i_j, \Delta_i, \Delta^i_j \rangle$, where j denotes other agents; i.e., $(j \in I \setminus i)$. The estimates of joint state $\hat{\vec{s}}^i$ and the transition functions $\hat{P}^i$ and $\hat{P}^{ji}$ are initialized using the initial joint state and an a priori estimate of the transition function, respectively, both of which are common knowledge. All components of the indicator functions Δ_i, Δ^i_j are initialized to 0. The local action policy, $\hat{\pi}_i$, and estimates of the policies of other agents, $\hat{\pi}^i_j$, are initialized using the known ActionPlanner, the initial estimate of the transition function, and the known goal states.

Model Propagate and Update Rules

Borrowing terminology from estimation theory, the agent carries out 'propagate' and 'update' procedures at each time step to maintain reliable model estimates. During execution, agent i propagates its model estimates at each time step in the absence of any communications. Upon receiving new information through incoming communication, the agent uses this information to update its model.

Propagation comprises of two steps - ModelSelfPropagate and ModelOtherPropagate (see Algorithms 1-2). In ModelSelfPropagate, the agent updates knowledge about its local state based on local observations, and updates $\langle \hat{P}^i, \Delta_i \rangle$ according to the available information based on the domain-dependent function ObserveDomain. For all the (s, a, s') tuples observed via ObserveDomain, the indicator function Δ_i is set to 1. Thus, Δ_i quantifies the amount of knowledge gathered regarding the unknown, deterministic domain. Additionally, ModelSelfPropagate provides a binary output, b_o, which quantifies whether or not the agent received any new information about the environment; b_o is set to 'true' if the estimate $\hat{P}_i$ changes from its past value.

Propagation of estimates corresponding to another agent j - i.e., $(\hat{s}^i_j, \hat{P}^{ji}, \hat{\pi}^i_j)$ - is carried out using ModelOtherPropagate. This function additionally requires as input information about whether or not agent j communicated during the previous time step. In the event that agent j did not communicate, then agent i first propagates the estimate of j's policy, $\hat{\pi}^i_j$. This is done by comparing the expected local reward of $\hat{\pi}^i_j$ denoted[2] as $E_{\hat{P}^{ji}}[\Xi_j(\hat{\pi}^i_j)]$, with that of a policy ψ^i_j after re-planning, and choosing the latter if it results in a greater reward. Policies of the agents that communicated during the previous time step are known based on their communication, and do not need to be recomputed during the propagation step. The policy estimate is used to compute the previous action based on the previous estimate of the agent state, which is then used to estimate the current state. Lastly, agent i updates its estimate of the transition function maintained by agent j, $\hat{P}^{ji}$, by using the SimulateObserveDomain function to simulate the transition function update of agent j. This function simulates ObserveDomain assuming the true transition function as $\hat{P}^i$.

Similarly, the update step includes two substeps: ModelSelfUpdate and ModelOtherUpdate (see Algorithms 4-5). Both these functions use MergeTransition, which merges the available information about the domain received from a sender $(\hat{P}^s, \Delta_s)$ into the receiver's model $(\hat{P}^r, \Delta_r)$. MergeTransition, described as Algorithm 3, updates a component of the transition function if and only if the corresponding component of the sender's indicator function is 1. This ensures that only observed information is used in the update. Further, in the event of a conflict between the two models for which $\Delta_s(\vec{s'}|\vec{s}, \vec{a}) = 0$, the receiver retains its previous estimate. Note that because we model one-way communication, only the receiver's estimate $\hat{P}^r$ and indicator function are updated. Using the MergeTransition function, the ModelSelfUpdate and ModelOtherUpdate incorporate the received information in the agent's model.

Algorithm 1 ModelSelfPropagate

```
function ModelSelfPropagate(M_i)
    b_o = false
    s_i ← obtained from local observations ;
    ⟨P̂^i, Δ_i⟩ ← updated based on ObserveDomain;
    if change in the estimate P̂^i then
        b_o = true ;
    end if
    return M_i, b_o
end function
```

Algorithm 2 ModelOtherPropagate

```
function ModelOtherPropagate(M_i, j, C_j)
    if C_j = 0 then
        ψ̂^i_j ← ActionPlanner( ŝ^i_j, P̂^ji )
        if E_{P̂^ji}[Ξ_j(π̂^i_j)] < E_{P̂^ji}[Ξ_j(ψ̂^i_j)] then
            π̂^i_j ← ψ̂^i_j
        end if
    end if
    â^i_j ← π̂^i_j(ŝ^i_j) ; ŝ^i_j ← arg max_{s_j} P̂^i_j(·|ŝ^i_j, â^i_j)
    P̂^ji ← SimulateObserveDomain(P̂^ji, ŝ^i_j, P̂^i)
    return M_i
end function
```

[2]The subscript of the expected value denotes the transition function used to calculate the expected reward, which in this case is $\hat{P}^{ji}$.

Algorithm 3 MergeTransition

function MERGETRANSITION($\hat{P}^r$, Δ_r, $\hat{P}^s$, Δ_s)
  for $(\vec{s}, \vec{a}, \vec{s'} \in S \times A \times S)$ **do**
    if $\Delta_s(\vec{s'}|\vec{s}, \vec{a}) = 1$ **then**
      $\hat{P}_r(\vec{s'}|\vec{s}, \vec{a}) \leftarrow \hat{P}_s(\vec{s'}|\vec{s}, \vec{a})$; $\Delta_r(\vec{s'}|\vec{s}, \vec{a}) \leftarrow 1$
    end if
  end for
  return $\hat{P}^r$, Δ_r
end function

Algorithm 4 ModelSelfUpdate

function MODELSELFUPDATE(M_i, j, $\hat{P}^j$, s_j, π_j, Δ_j)
  $\langle \hat{P}^i, \Delta_i \rangle \leftarrow$ MergeTransition$(\hat{P}^i, \Delta_i, \hat{P}^j, \Delta_j)$
  $\langle \hat{P}^{ji}, \hat{s}^i_j, \hat{\pi}^i_j \rangle \leftarrow \langle \hat{P}^j, s_j, \pi_j \rangle$
  return M_i
end function

Algorithm 5 ModelOtherUpdate

function MODELOTHERUPDATE(M_i, j,$\hat{P}^j$,Δ_j)
  for $(k \in I \setminus i)$ **do**
    $\langle \hat{P}^i_k, \Delta^i_k \rangle \leftarrow$ MergeTransition$(\hat{P}^i_k, \Delta^i_k, \hat{P}^j, \Delta_j)$
  end for
  return M_i
end function

Making Replanning and Communication Decisions

The model of the agent obtained after the 'update' and 'propagate' steps is used to make the re-planning and communication decisions. If an agent receives novel observations, as indicated by b_o, it has to decide whether to use this information for re-planning, communicating, both or neither. Replanning its own actions without communicating may lead to a loss of coordination, while communicating each time prior to using the information may result in high communication costs. Lastly, although not using observed information does not lead to loss of coordination or any communication cost, it may result in poor task performance due to use of a stale model and plan. Thus, either option available to the agent can potentially be the best decision, depending on the domain and the problem state. To determine which decision is optimal, the communicating agent must use its current model to assess the impact of utilizing information unavailable to other agents. To compare the benefits of the available choices, we define the following three quantities of interest that must be estimated after gathering new observations about the world:

- α, the team reward estimated to result from the previously chosen policy if it is executed within the updated world model. This may not be identical to the true reward, but is the best estimate that the agent can calculate with the available information. Given that the agent does not modify its policy, coordination if present before is maintained.
- β, the team reward estimated to result from modifying the policy locally but not communicating this modification to other agents. This may result in better performance from the agent; however, by not communicating the information, the agent is at risk of poor coordination within the team. The agent must calculate the potential gain in reward from using the local information, and the potential reduction in reward due to the possible poor coordination. However, no communication costs must be factored in.

Algorithm 6 The ConTaCT algorithm

1: **function** CONTACT($M_i(t-1)$, $C_j(t-1)\ \forall j \in I$ and $\langle \hat{P}^j, s_j, \pi_j, \Delta_j \rangle \forall j \in \{I$ such that $C_j(t-1) = 1\}$)
2: Calculate total number of communications within the team since the previous time step: $N_c = \sum_{j \in I} C_j$
3: $\forall j \in \{I \setminus i$ such that $C_j = 1\}$: $M_i \leftarrow$ ModelSelfUpdate $(M_i, j, \hat{P}^j, s_j, \pi_j, \Delta_j)$
4: $\langle M_i, b_o \rangle \leftarrow$ ModelSelfPropagate (M_i)
5: $\forall j \in \{I \setminus i\}$: $M_i \leftarrow$ ModelOtherPropagate(M_i, j, C_j)
6: $\forall j \in \{I$ such that $C_j = 1\}$: $M_i \leftarrow$ ModelOtherUpdate$(M_i, j, \hat{P}^j, \Delta_j)$
7: **if** $N_c = 1$ **then**
8: $\forall j \in \{I \setminus (i, C_j = 1)\} : \hat{\pi}^i_j \leftarrow$ActionPlanner$(\hat{s}^i_j, \hat{P}^{ji})$
9: **else if** $N_c > 1$ **then**
10: $\forall j \in \{I \setminus i\} : \hat{\pi}^i_j \leftarrow$ ActionPlanner($\hat{s}^i_j$, $\hat{P}^{ji}$)
11: **end if**
12: **if** $b_o = true$ **then**
13: Calculate α, β, γ.
14: Select the maximum among α, β, γ. In case of ties, prefer α over β over γ.
15: If maximum is β or $\gamma : \pi_i \leftarrow$ ActionPlanner(s_i,$\hat{P}^i$)
16: If maximum is $\gamma : C_i \leftarrow 1$
17: **end if**
18: **return** $M_i(t), C_i(t)$
19: **end function**

- γ, the team reward estimated to result from a globally modified policy, wherein an agent communicates an observation, and all agents then work with a modified policy. This includes the reward resulting from use of novel information and the communication cost; however, no costs due to poor coordination must be factored in.

After receiving new observations, the agent calculates the above three quantities and selects the option which results in maximum expected reward among the three. To avoid redundant communication, the agent performs this computation if and only if its local observations contain novel information regarding the domain. In case of ties between α, β and γ, preference in choosing the maximum is given in the order (α, β, γ). This order is chosen to reduce the amount of re-planning and communication in the team. The agent communicates if γ is the maximum. The agent re-plans its actions if the maximum is either β or γ. Upon receiving a new communication the agent updates its model and re-plans its actions.

Algorithm 6 presents ConTaCT, which is called by each agent at each time step. The algorithm includes the model propagate and update steps, and the logic for deciding whether to communicate and/or re-plan. The algorithm takes as input agent i's current model as of the previous timestep $M_i(t-1)$ and the incoming communications since the previous timestep, $C_j(t-1)\ \forall j \in I$ and $\langle \hat{P}^j, s_j, \pi_j, \Delta_j \rangle \forall j \in \{I$ such that $C_j(t-1)$=1$\}$. The algorithm outputs agent i's updated model $M_i(t)$ and its communication decision $C_i(t)$. In line 2, the agent i first computes the number of communications within the team since the previous time step, N_c. In line 3-4, the agent uses the received communications (from agents with $C_j = 1$) and local observations to improve its model M_i. In line 4, the agent also computes b_o, which indi-

cates whether new information regarding the environment is observed. Next, the agent propagates its estimate of the transition and indicator functions maintained by other agents (line 5), and incorporates the effect of communication on them via ModelOtherUpdate (line 6). The agent then recomputes plan estimates $\hat{\pi}_j^i$ using the updated model (lines 7-11). In case of only one sender ($N_c = 1$), agent i recomputes its plan estimates for other agents except for the sender (who does not receive any new communication). When there are multiple senders ($N_c > 1$), agent i recomputes plan estimates for all the other agents, since all receive new information and initiate replanning. Lastly, in lines 12-17, if the agent receives any novel information ($b_o = 1$) it makes replanning and communication decisions based on the parameters (α, β, γ).

Results

We empirically evaluate the efficacy of ConTaCT through simulations of a multi-agent task motivated by rescue operations during disaster response scenarios. In this section, we briefly describe the simulated domain, the computational policies against which we benchmark our algorithm and the results of our simulation experiments.

Task Description We consider a hypothetical disaster response scenario in which a team of first-responders answers a rescue call. At the outset, the responders are distributed at known starting locations, and the location of the person to be rescued is known. We model the environment as a grid world with obstacles. Decision making is fully decentralized and the action planners used by each agent are known and identical. For this scenario, action planning corresponds to path planning and ActionPlanner is chosen as a single agent path planner (Likhachev, Gordon, and Thrun 2003). A map of the environment is available; however, it does not reflect any changes to the environment resulting from the disaster. This requires the responders to operate within a potentially unknown environment. During execution the agents can observe their own state but not that of any other agent. Further, the ObserveDomain function of our task model corresponds to the agents being able to obtain true information regarding the adjacent grids of the map during execution.

During the rescue operation, the responders have an option to communicate with one another. Each communication consists of an agent sharing their current, labeled map of the environment (which may differ from the initial map due to novel observations) with labels indicating whether or not they know a part of the map accurately, their location and trajectory, and their belief about location of each agent. The communication takes up to one time step to reach other agents. Further, there is a pre-specified cost associated with a communication ($-R_{ic}$). The objective of the responders is to reach and rescue the person before a pre-specified deadline while maximizing the team reward. The task is successfully completed if and only if all the responders reach the rescue location before the deadline.

Experiment Details The task is modeled as a TCD-DEC-MDP-COM with unknown transitions. Agents incur an action reward, R_{ia}, of -1 in all states but for the goal state, and terminal reward, R_{ig}, of -10^6 if they do not reach their goal by the deadline (time horizon). The chosen simulated scenario is analogous to the benchmark problem Meeting in a Grid (Bernstein et al. 2002); however, in our evaluation action and sensing uncertainty is absent, and agents have the additional challenge of reasoning with imperfect knowledge of the transition function (map). We use a larger map and team size in comparison to the prior work.

We evaluate the performance of ConTaCT on randomly generated grid worlds with varying communication reward, R_{ic}. One fifth of the grids, on average, are labeled as obstacles for the ground truth map of the environment. Each agent's initial knowledge of the map is imperfect; 30% of grids, on average, are mislabeled either obstacle or free space for the agents. The initial and goal states are common knowledge and are sampled from a uniform distribution over the obstacle-free grid squares.

The performance of ConTaCT is benchmarked against two policies, no communication (NC) and re-plan only on communication (RooC). The RooC baseline is implemented by eliminating the β alternative from ConTaCT, and modifying the propagate rule to reflect no communication implies no re-planning. The RooC baseline is motivated by the algorithm DEC-COMM (Roth, Simmons, and Veloso 2005) designed for DEC-POMDPs with known transition functions. In DEC-COMM agents do not use local information without communicating it, and communicate only if the expected reward of the new joint action post communication is higher than that of the current joint action.

Discussion Table 1 summarizes the experimental results. While average team reward is comparable for the tasks in which all algorithms resulted in sucessful completion, teams completed tasks successfully marginally more often with RooC than with ConTaCT or NC. For instance, for the fifty trials on a 10×10 grid with five agents and $R_{ic} = -1$, teams using RooC succeeded in 39 tasks as opposed to the 35 and 26 tasks completed by teams using ConTaCT and NC, respectively.

However, this marginal improvement in success rate of RooC as compared to ConTaCT comes at the cost of significantly higher numbers of communication messages. Agents using RooC always communicate prior to using new information, resulting in a marginally higher success rate but many redundant communications. Teams communicated only 43 times using ConTaCT, as compared to 112 for RooC, a more than two-fold difference. Similar trends are observed across problems with different grid sizes, number of agents and communication cost. Teams using ConTaCT achieve comparable performance with more than a 60% reduction in the number of communications.

ConTaCT's comparable performance despite the small number of communications is possible since each agent is able to use information locally without necessarily communicating the same. This is advantageous when the information benefits only the local agent but not necessarily other members of the team. Thus, the agent communicates if and only if the information benefits the team and thereby maintains similar task performance with less number of communications.

Table 1: Summary of Simulated Results.

					ConTaCT		Replan only on Communication		No Communication	
Grid Size	Agents	Time Horizon	Trials	$-R_{ic}$	Successful Tasks (%)	Total # Comm. across all trials	Successful Tasks (%)	Total # Comm. across all trials	Successful Tasks (%)	Total # Comm. across all trials
5×5	2	25	50	0	84	9	86	46	84	0
			50	1	84	5	86	45	84	0
			50	2	84	5	86	43	84	0
10×10	5	100	50	0	70	43	78	112	52	0
			50	1	70	32	78	95	52	0
			50	2	70	29	78	93	52	0
25×25	5	625	20	1	35	21	40	53	35	0

This behavior is especially desired in human-robot teams, where excessive communication from an agent may hinder the human's task performance.

Since, the ConTaCT algorithm requires the agent to communicate and maintain estimates of its transition function, as opposed to the observation history, the memory requirements of the algorithm are fixed. For implementation in real systems, protocols may be designed that require the agents to communicate only the difference between the current transition function and previous common knowledge to efficiently use the available communication bandwidth. Lastly, we note that the ConTaCT algorithm provides a general approach to making communication decisions through the consideration of parameters (α, β, γ) and can work with definitions of team reward other than the one specified by our task model.

Conclusion

In this paper, we present a novel model, TCD-DEC-MDP-COM, for representing time-critical collaborative tasks in deterministic domains. This is motivated by applications, including disaster response and search and rescue, where the outcome of agents' actions can be modeled as certain but the environment is often initially unknown. We develop an algorithm, ConTaCT, that generates re-planning and communication decisions for tasks modeled as a TCD-DEC-MDP-COM with unknown transitions. Simulated experiments are conducted for hypothetical rescue tasks. Results suggest that ConTaCT has the potential to substantially reduce communication among agents without substantially sacrificing performance in the task.

Acknowledgments

We thank Chongjie Zhang for useful discussions.

References

Amato, C.; Konidaris, G.; How, J. P.; and Kaelbling, L. P. 2014. Decentralized Decision-Making Under Uncertainty for Multi-Robot Teams. In *the Workshop on Future of Multiple Robot Research and its Multiple Identitie at IROS*. IEEE.

Amato, C.; Konidaris, G. D.; and Kaelbling, L. P. 2014. Planning with macro-actions in decentralized POMDPs. In *AAMAS*.

Amir, O.; Grosz, B. J.; and Stern, R. 2014. To share or not to share? the single agent in a team decision problem. In *AAAI*.

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2004. Solving transition independent decentralized Markov decision processes. *JAIR* 423–455.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27(4):819–840.

Goldman, C. V., and Zilberstein, S. 2003. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS*.

Kamar, E.; Gal, Y.; and Grosz, B. J. 2009. Incorporating helpful behavior into collaborative planning. In *AAMAS*.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2003. ARA*: Anytime A* with provable bounds on sub-optimality. In *advances in NIPS*.

Mostafa, H., and Lesser, V. 2009. Offline planning for communication by exploiting structured interactions in decentralized MDPs. In *International Conference on Intelligent Agent Technology*, volume 2, 193–200.

Nair, R.; Roth, M.; and Yokoo, M. 2004. Communication for improving policy computation in distributed POMDPs. In *AAMAS*, 1098–1105.

Pynadath, D. V., and Tambe, M. 2002. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *AAMAS*, 873–880.

Roth, M.; Simmons, R.; and Veloso, M. 2005. Reasoning about joint beliefs for execution-time communication decisions. In *AAMAS*.

Spaan, M. T.; Gordon, G. J.; and Vlassis, N. 2006. Decentralized planning under uncertainty for teams of communicating agents. In *AAMAS*, 249–256.

Williamson, S.; Gerding, E.; and Jennings, N. 2008. A principled information valuation for communications during multi-agent coordination. In *Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains at AAMAS*.

Williamson, S. A.; Gerding, E. H.; and Jennings, N. R. 2009. Reward Shaping for Valuing Communications During Multi-Agent Coordination. In *AAMAS*, 641–648.

Wu, F.; Zilberstein, S.; and Chen, X. 2011. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence* 175(2):487–511.

Xuan, P.; Lesser, V.; and Zilberstein, S. 2004. Modeling Cooperative Multiagent Problem Solving as Decentralized Decision Processes. In *AAMAS*.