Thesis

Joris Z. van den Oever

March 7, 2016

Abstract

To be written

Contents

1 Introduction **1**

2 Problem Statement **1**
2.1 Research Questions 2
2.2 Research approach 2

3 Experiment Design **3**

4 Analysis **6**

5 Mitigation Strategies **7**

6 Future Work **7**

A Goal Agents Source **7**
A.1 No Communication Agent 7
A.2 Common Code 10
A.3 Preliminary research data 12

1 Introduction

2 Problem Statement

Robotic systems have been getting more mobile and prevalent in recent years. And with that comes new possibilities and uses. One project trying to take advantage of this is the TRADR project, Long-Term Human-Robot Teaming for Robot-Assisted Disaster Response[?]. It aims to create teams of robots and humans that work together to provide incidence response. Research into robot team communication is what prompted this research.

Autonomous robots in in disaster areas, including the industrial areas given by the TRADR use cases, need to be able to cooperate amongst themselves and with humans[?]. However because of the mobile nature of such robots this raises new questions about what happens when communication becomes unreliable. Wireless signals may not reach everywhere[?, ?] and wired communication isn't generally feasible when staying mobile.

This requires us to look closer at the effects of having communication fail. While it is easy to presume that this will have undersired effects it is hard to predict what effects that will be. Only once these effects, if they exist, have been measured we can look at potential mitigation strategies. And of course determine their effectiveness.

2.1 Research Questions

To this end we will provide answers to the following questions:

- What are the effects of communication failure of multi-agent team effectiveness?
- If there are negative effects to the effectiveness, how can we mitigate these effects?

Here we define a single instance communication failure as the recipient not having a message that the sender meant for it to have. Irregardless of where in the transmission process the failure occured, e.g. while sending or receiving. And effectiveness is both the rate of task completion, and the time taken to complete the task.

2.2 Research approach

To gather the data we will create simulations of agent teams enduring communication failure. How this is set up exactly is descriped in section 3. The differences will be used to see if the effects differ depending on how the communication fails and how well different agent strategies deal with the failure. This will all be simulated as it allows for stricter control of the experimental dimensions in addition to making it easier to perform a large amount experiment runs for each scenario. Afterwards the results of the simulations can be compared to baselines where no communication failure is happening to answer the first research question. Then based on the data we can see where the effects are strongest, and if there are agents that work well in different situations. From these we can identify potential mitigation strategies. Which can be simulated in the same scenarios as before to compare their effectiveness and answer the second question.

The gathered data will be analysed using Chi square analysis to determine if the results are significantly different from the reference data. This will give us two analyses using both the one agent team baseline and the no communication

team. This comparison we will only look at the rate of task completion within the timeout period.

If this shows there are indeed significant differences we can look for the effect the different communication failure scenarios on task completion time. As a first step regression analysis can see if there is an effect using both the type of team, size of the team, and the failure scenarios as independent variables. Pending determination on if they have a signicant effect on the task completion rate. As an additional analysis to if there is a difference in how the different team types are affected each team type will have a separate regression analysis using the team size and failure scenarios. The same analysis will be done using the task completion time as the dependent variable.

3 Experiment Design

To determine the effects of communication failure, if any, on multi-agent systems we will use a well explored domain for multi-agents systems. The Blocks World for Teams GOAL environment as this is well defined and thoroughly explored system for doing research on multi-agent systems.(references) The basic concept is that an agent, or multiple agents, have to bring blocks to a certain place, the dropzone in a specific order. The blocks are distinguished by color only so any block that is the right color will suffice. These blocks are found in rooms that have one entrance from the hallway spaces. While that is the basic scenario it has built in several option for adjusting various environment variables such as the room placement

Experiment Dimensions

However it is not just the room placement that can be varied. We have categorised the variables involved in the above mentioned experimental setup. Each categorgy has multiple dimensions as shown in Table 1 and will be discussed in more detail.

The environment is mainly the way the experiment world is constructed, how many hallways and where, and the placement of the rooms, but also includes the block sequence and agent collisions.

Second is the agents and their teams themselves. The team size can be adjusted, limited by the amount of computing power available. Though world limitations, such as only one agent can enter a room at a time, might limit the effectiveness of a team larger than the amount of rooms available or than there are blocks in the sequence. Once you have a team you can consider team composition. Are all agents the same or do they have different programs. Finally, what kind of programs do the agents run with. Minimal programs, full coordination, what kind of data do they share, which resolution strategies are used, there is an almost infinite amount of possibilities here.

Finally there is the communication and how it can fail. Does the sending or receiving fail, and if it does fail is it a fixed percentage of messages that fails?

Environment	World size and room placement	Block sequence	(No) Agent collisions
Agents	Team size	hetero-/homogenous teams	Agent type
Communication	Send/receiving failure	Failure chance	(No) global percepts

Table 1: Experiment Dimensions

Of does the failure rate increase over time? is the message more likely to fail if the receipient is further away? Those are several ways the failure of messages can be done. Finally there is something to consider regarding communicating itself. Do the global percepts provided by the environment always arrive? Since they include things like when a block is delivered at the dropzone. This could effectively be a side-channel for certain kinds of messages that always succeeeds. So the agents can not use it and have to distribute such information through the normal channels.

Experiment reduction

The above mentioned experiment dimensions have too many options to realistically test everything. However we have evaluated all of them to see where the dimensions can be reduced so it only requires a manageable amount of simulations. The dimensions are discussed grouped by category as given by table 1.

Environment

The size and shape of the world the agents move around in for example can be kept consistent and seen as outside of the scope for these experiments. As that would be looking at the effects of the environment on communication failure. As such only one map design will be used across the experiments.

However switching the sequence between random and only one colour is still a possibility left to us. However all the same colour, would eliminate the need to communicate about where in the sequence the team is. Which, for some types of agents, is the only thing communicated. As such using random, where it can not be assumed the communication isn't needed, is the preferred case. While the simplicity that can be assumed with just one colour might help in building the agents the increased communication if that isn't the case is preferred.

Agents

Limiting the number of sizes used for the agent teams would drastically reduce the experiment space as well. And not all sizes need to be tested to get a sense of the effects, if there is one, communication failures might have. The case of having only one agent, would not need a lot of experimentation as there is nothing to communicate with, however it will still be useful to do once as a baseline reference. Next, using a size sample of three agents will be useful to test, as this is the minimum amount of agents needed to have a team where it

is possible to communicate with more than one agent at the same time. Which is needed for some communication failure modes to matter. Next would be five agents, as this is a larger group, but is also just a bit more than half the amount of rooms available in the default map setup and one less than the amount of blocks needed in the sequence. This means the way tasks get divided amongst agents becomes important. As not every agent can continue exploring after just one set of rooms. As well as increasing the amount of agents involved in communicating. The next step should not go too far, as experience with the environment has shown that 10 agents start to tax the hardware and might degrade the simulation. But looking at a group with a size close to the amount of rooms available will be interesting to see how that influences the behaviour. As such going one agent fewer than the amount of rooms in the standard setup, eight, has been chosen as the last group size.

All these agent teams still need a team composition. While they could have different programs, this might introduce higher order effects that would make analysis more complicated, as well as requiring a lot more simulations. While these effects might be interesting, determining the effects, if any, in the simple case takes priority. So in the interest of limiting the time spent with simulations, and analyzing homogenous agent teams will be used.

But that still leaves options for varying the teams themselves, though they will all have similar components as shown in figure 1. These agent programs are chosen from all possibilities because they are simple, build upon each other, and let us explore what happens with differing degrees of communication.

To properly compare the teams baselines are needed. The simplest baseline is a single agent exploring rooms and delivering blocks as soon as it finds the currently needed block. This will let us evaluate how teams perform compared to a single agent, even with the communication failure. This is then used to make the team baseline, which only communicates which blocks have been delivered. A third team will also communicate which blocks can be found in rooms they explore.

From here the agent types branch out into two types that differ in their coordination strategies. The agents will coordinate their actions to see who is best suited to do what task at any given moment. But the way they coordinate differs, one team asks first if they can do something, before conflict resolution happens if needed. The other team starts doing a task, and then verifies that nobody else is doing it, before going to conflict resolution.

Communication

For the experiment it is pretty important to figure out what kind of communication failure is going to be tested. The simplest of which is to test the difference between sending failure and receiving failure. As sending failure means that if a message fails no agent gets it but receiving failure means that some part of the team may receive it. So both options will be tested.

This can be done with various ways of determining the likelihood that a message will fail. For reference each team will also do simulations without com-

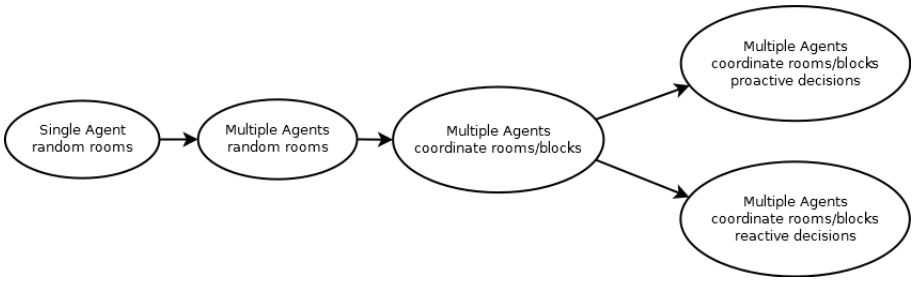

Figure 1: Agent types, each agent team has the capabilities of the previous one in the graph

munication failure, meaning 0%. To get insight into how different amounts of failure affect performance using set percetages is very useful. This will be a spread from 0 to a100%. To limit the amount of runs needed 25% increments will be used initially. However all communication and no communication could have different results from only a little or a lot of communication 5% and 95% will be tested as well. Making the complete range of tested percentages $0, 5, 25, 50, 75, 95$, and 100%

Finally, how are global percepts treated. These function similar to a second, infallible communication channel. As this is all about testing the effects of communication failure these will be ignored. Instead for the current state of the sequence has to be communicated. But to prevent complete breakdown given the fixed percentage failures the agents can still see the state of the sequence when they enter the dropzone. Allowing for some corrections as preliminary testing with already existing agents has shown that without the correction programs break down fast. Even a 5% failure can cut the amount of successful runs in half. An overview of these results can be found in table 3.

4 Analysis

Table 2 gives us an overview of what was expected to happen in the various scenarios. While even the guaranteed successful scenarios might not finish within the timeframe given for the experiments they should more often do so than in scenarios where success might not always be possible. In these cases different strategies can be compared for how much they slow down compared to the reference and what the major causes for slowdown are.

Success Guaranteed	Could fail
-	sequenceUpdate broadcast fails: wrong blocks de
room/block information broadcast fails: slows down	-
Agent Intention Broadcast: slows down	-
-	Agent plan negotiation message failure: Might wait fo

Table 2: Table of the effects communication failure might have. On the left success is guaranteed if given time. On the right situations may arise where success is no longer possible.

5 Mitigation Strategies

6 Future Work

References

A Goal Agents Source

The agent teams each have their own main module. However a lot of the other modules are shared as they share capabilities. First the different agents are detailed. Then the shared modules and knowledge.

A.1 No Communication Agent

This agent does not communicate and serves as the basis for both the single agent team and the no communication multiple agents team. The only difference between the two is the amount of agents instantiated by the .mas file.

```
1  use robot as knowledge.
2  use robot as actionspec.
3  use updateSequence as module.
4  exit=nogoals.
5  module main {
6          %drop blocks in the dropzone when they are needed
               , and communicate this.
7          if bel(in('DropZone'), holdingNextBlock) then
               putDown + updateSequence.
8
9          %drop blocks in rooms when they are not needed
               anymore
10         if bel(in(Loc),Loc\='DropZone', not(
               holdingNextBlock)) then putDown.
11
12         %go to goal places
```

```
13          if a-goal(in(Place)) then goTo(Place).
14
15          %go to goal blocks
16          if a-goal(holding(BlockID)), bel(in(Place), block
                (BlockID,_,Place), not(atBlock(BlockID))) then
                 goToBlock(BlockID).
17
18          %pickup goal blocks
19          if a-goal(holding(BlockID)), bel(atBlock(BlockID)
                ) then pickUp.
20  }
```

robotEvents.mod2g

```
1   use robot as knowledge.
2   use robotGoals as module.
3   use updateSequence as module.
4
5   module robotEvents {
6           %-------communication updates & conclusions about
                 other agents-------
7           forall bel( send('allother', Y) ) do allother.
                send(msg(Y)) + delete(send('allother', Y)).
8
9           % deduce information based on deliveries of other
                 agents:
10          forall (Agt).sent(msg('deliveryDone')) do
                updateSequence.
11
12          % Update the agent's state of movement.
13          forall bel( state(State)), percept(state(NewState
                )) do delete( state(State) ) + insert( state(
                NewState) ).
14
15          % Record when we are entering or leaving a room.
16          forall percept(in(Place)) do insert( in(Place) ).
17          forall percept(in(Place)), bel( not(visited(Place
                )), room(Place) ) do insert( visited(Place) ).
18          forall percept(not(in(Place))) do delete( in(
                Place)).
19
20          % Discover new blocks
21          forall percept(color(BlockID, ColorID)), bel( in(
                Place), not(block(BlockID, ColorID, Place)) )
                do insert( block(BlockID, ColorID, Place) ).
22
```

```
23          % Record atblock location of agent
24          forall percept(atBlock(BlockID)) do insert(
                atBlock(BlockID)).
25          forall percept(not(atBlock(BlockID))) do delete(
                atBlock(BlockID)).
26
27          % Record if a block is being held
28          forall percept(holding(BlockID)) do insert(
                holding(BlockID)).
29          forall percept(not(holding(BlockID))) do delete(
                holding(BlockID)).
30
31          %remove blocks that are not held or in the room
                anymore
32          forall bel(in(Place), block(BlockID, ColorID,
                Place), not(holding(BlockID))), not(percept(
                color(BlockID, ColorID))) do delete(block(
                BlockID, ColorID, Place)).
33
34          %remove blocks that are not held or in the room
                anymore
35          forall bel(in(Place), block(BlockID, ColorID,
                Place), not(holding(BlockID))), not(percept(
                color(BlockID, ColorID))) do delete(block(
                BlockID, ColorID, Place)).
36
37          % Update sequence when in dropzone.
38          if percept(sequenceIndex(X)), bel(sequenceIndex(
                OldX)) then delete(sequenceIndex(OldX)) +
                insert(sequenceIndex(X)).
39          if bel( in('DropZone'), seqDone(Seq), length(Seq,
                 N), sequenceIndex(X), N < X , sequence(
                NewSeq), length(NewSeq, X), append(NewSeq, _,
                Full) ) then delete( seqDone(Seq) ) + insert(
                seqDone(NewSeq) ).
40
41          %remove obsolete goals
42          if goal(holding(BlockID)), bel(not(block(BlockID,
                 ColorID, Place))) then drop(holding(BlockID))
                .
43
44          %adopt new goals (and stop traveling)
45          if not(goal(in(Place))), goal(seqDone(_)) then
                adoptgoals.
46  }
```

A.2 Common Code

robot.act2g:

```
use robot as knowledge.

% The goTo action makes the agent move to a place (location) in the BW4T environment.
% As long as the agent has not arrived at the place it is going to, it will be in "traveling" mode.
define goTo(Location) with
        pre { not(state(traveling)) }
        post { true }

% goToBlock only when not traveling and in a room
define goToBlock(BlockID) with
    pre {in(_), not(state(traveling)) }
        post { true }

% pickUp can only be performed when not holding a block
define pickUp with
        pre{not(state(traveling)), not(holding(_))}
        post{ true }

% putDown can only be performed when holding a block
define putDown with
        pre{not(state(traveling)), holding(_)}
        post{ true }
```

UpdateSequence.mod2g

```
use robot as knowledge. order=linearall.

module updateSequence{
        %update sequence
        if bel(seqDone(SDone), nextNeededColor(ColorID),
            append(SDone,[ColorID],NewSDone) ) then delete
            (seqDone(SDone)) + insert(seqDone(NewSDone)).

        %remove beliefs about the delivered block (if
            this agent was delivering it)
        if bel(in('DropZone'), holding(BlockID), block(
            BlockID, ColorID, Place)) then delete(block(
            BlockID, ColorID, Place)).
}
```

robotGoals.mod2g

```
1  use robot as knowledge. order=linear.
2
3  module adoptgoals{
4          %If holding the next needed block go to the
               dropzone.
5          if bel( holding(BlockID), nextNeededColor(ColorID
               ), block(BlockID, ColorID, _)) then adopt(in('
               DropZone')).
6
7          %Otherswise, If the next needed block is known
               then adopt a goal to go there and hold it.
8          if bel( nextNeededColor(ColorID), block(BlockID,
               ColorID, Place)) then adopt(in(Place),holding(
               BlockID)).
9
10         %Otherwise go to a random room we haven't seen
               yet'.
11         if bel( not(finished), bagof(Place, (room(Place),
                not(dropZone(Place))), Places), random_member
               (Dest, Places) ) then adopt(in(Dest)).
12 }
```

robotInit.mod2g

```
1  use robot as knowledge.
2
3  module robotInit {
4          % Store map information, i.e., navigation points
               in the agent's belief base.
5          forall percept(zone(ID, Name, X, Y, Neighbours))
               do insert( zone(ID, Name, X, Y, Neighbours) ).
6
7          % Record the initial state of movement in belief
               base.
8          if percept(state(State)) then insert( state(State
               ) ).
9
10         % Record goal sequence
11         if percept( sequence(Seq) ) then insert(sequence(
               Seq), seqDone([])) + adopt(seqDone(Seq)).
12         if percept( sequenceIndex(X) ) then insert(
               sequenceIndex(X) ).
13
14         % Adopt initial goal going to a random place
15         if bel(room(PlaceID), PlaceID\='DropZone') then
               adopt(in(PlaceID)).
```

Agents\chance of failure	reference	cuttoff	5%	10%	25%	50%	95%	distance	50% send failur
Blockbuster Mincom	193	101	109	98	74	64*	63		45*
Blockbuster dzone only	199	11	98	50	15	5	16		
Blockbuster all room	190	18	85	33	13	52	38		60*
Maximum Coordination	188	168	182	182	162	138	64		

Table 3: The amount of successes over 200 runs per agent per failure mode. All agent and failure mode combinations resulted in a chi square $p < 0.01$ except for Coordination at 5%($p \approx 0.07$) The entries marked with * have been performed with fewer than 200 runs and scaled up to the same amount so less accurate.

```
16 }
```

A.3 Preliminary research data