# Improving Agent Team Performance through Helper Agents

Marie D Manner and Maria Gini

Department of Computer Science and Engineering, University of Minnesota
manner@cs.umn.edu, gini@cs.umn.edu

**Abstract.** In this paper, we consider the problem of environmental constraints on teams and examine how the constraints impact teamwork. We simulate a realistic problem for teams in the physical world by implementing a limit on the range of communication among agents. Using BlocksWorld for Teams from Delft University of Technology, we build teams of 3-4 agents with varying communication ranges and of different agent type (regular task-performer, communication-only or reconnaissance-only) to test team performance in the constrained environment. We analyze the results and discuss implications to the team mental model. We show that adding a helper agent to the team can reduce task completion time, but not for all types of agents and environments.

## 1 Introduction

Imagine a dangerous, rocky terrain, strewn with unexploded bombs or traps. On one side of the field lies a civilian camp in need of food and medical supplies; on the other, a force with aide supplies to help. The aide team sends out a small deployment of robots to investigate and deactivate the bombs, possibly mapping the terrain or positioning additional hardware in the field. The agents may fail due to hardware error, such as battery drain or slow travel speeds, or software malfunction. In such a case, robots are surely more expendable than humans. They could coordinate amongst themselves, split the tasks, and return when finished, without any real human supervision. For such a team to work well, it needs to communicate in the face of hardware failures, function after failures of other team members, and recognize when parts of the goal have been completed. We use this scenario to motivate our research with teams in imperfect environments.

Teamwork is a well-known force for efficiently solving tasks. A team is different than simply a group of people – good teams have coordinated action, mutual understanding, high commitment, role differentiation, and shared goals [24]. Team members must understand the technology used, share job or task models, and hold shared concepts of how the team interacts [15]. Sharing relevant information and task knowledge, be it techniques and methods of doing or resource locations, helps each team member perform his own task faster and predict what support his team members might need and give it to them faster or even before they ask.

To build a team that can work together smoothly and effectively, we should model the structure of team information and action process in what is known as a "team mental model," which includes communication, understanding, shared goals, technological

abilities, intentions and need prediction [3, 11, 14]. When we develop our team in Section 4, those are the basic six components we consider. A team mental model is the particular schema used by each agent in a team, whether the agent is human or robot; as an agent experiences the world, its personal mental model will become populated with data or beliefs. The better the team mental model, the more easily agents can understand each others' actions or interests, enabling the agents to interact, cooperate, and run more smoothly as a team. Team members share ideas or information about their task, and each member adjusts his or her task based on what other team members will do, have done, or can't do. One team member's equipment breaks and it asks for help; another team member picks up the slack. One team member does the first small task in the goal, and other team members do not need to verify the task is done – they go on to other jobs to finish the team goal faster.

Now imagine that you have the perfect team. Members communicate, help each other, and accomplish the goal in front of them quickly and efficiently. What happens when you take the perfect team and stick it in an imperfect environment? Radio equipment breaks down, agent hardware or software inexplicably fails, and communication fails due to natural causes or interferences. How does our team deal with such an imperfect, and more realistic, environment? Our goal is to show that we can make teams of agents with imperfect communication abilities perform as well as (or very close to) teams with perfect communication abilities.

The main contribution of this paper is an experimental study which demonstrates that adding various agent types to a team in a constrained environment can improve team coordination up to the point of performing at the level of a team without those constraints. The context for our work is small teams engaged in an exploration task. We show how the presence of a communicator or reconnaissance agent affects team performance for different communication ranges.

We review related work in Section 2 and give an overview of the problem along with some background in Section 3. We review our experimental approach and results in Section 4, future work in Section 5, and conclusions in Section 6.

## 2 Related Work

The work we present builds on a large body of knowledge, which spans from mental models of team members to team structures used in human teams.

We first consider how to approach the concept of a team. Paris et al. [16] reviewed research in understanding human team performance and found no less than eight representative theories on teamwork. Such approaches examine, among other things, work-related, interdependence, capability, lifecycle evolution or maturation, and task-oriented implications of the team member interactions or relationships.

Of special relevance to our work is the study in [19] which highlights the differences between conventional and distributed teams. Distributed teams have a more flexible organization, but have a harder time in obtaining situational awareness. The lack of visual cues reduces their performance, affects their communication needs, and makes tightly coupled tasks harder to complete [7].

Knowing that we might model teams in different ways depending on things like team lifecycle or capabilities, we next address the issue of how to form a team mental model, which includes deciding which components we might include. Empirical contributions on the mental model of geographically distributed human teams are given in [5]. Work by Jonker et al. [13] considers components of a mental model and how to measure the sharedness of the model across agents. The team mental model resides in the mind, and is contrasted by any physical model. An agent has, along with its mental model, a physical model, goals, a team, a mind, an extension of the mind, and the actual system of concern. The authors argue that shared mental models apply to human-agent teams, and that agent designers can use the idea of mental models to improve teamwork.

Because we do not expressly want to leave humans out of our teams, we must consider how adding people may impact goal performance in the form of changing levels of motivation or commitment. Johnson et al. [10] investigated how human participants on a team with agents thought they worked better as a team when the agents provided explanations about their actions, even when the agents performed the same tasks regardless of human input. This belief may contribute to trusting other team members more, and this established trust helps the team work together better. When we consider human-agent teams, we should consider the happiness of the human team members, which may impact the team's task – unhappy or unmotivated humans may become indifferent to their goal, do less useful work, or sabotage the task.

Now that we have considered how to build a team and construct the mental model, we examine the practicality of human-robot communication and review some work on communicating in real-world situations. Related work shows us that human-robot interaction is feasible, and that if communication includes all facts and intentions, the team performs more efficiently. Jonker et al. [12] sought to measure the level of "sharedness" of a mental model and found that when agents communicate world and intention information, agent teams perform tasks faster. Rekleitis [17] investigated multi-agent teams with limited, line-of-sight communication, but focused on coverage of an entire area with agents. Other work considers intermittent communication or real-world problems, but uses controllers, requires networks, forms subteams, or requires close proximity [23, 9]. We would like our team to explore without requiring total coverage and without using a mastermind to direct branches of the team.

Previous research tells us we can safely model teamwork, training, and even use simulations for model development, so we now look at how we can make teams operate better. Sycara and Sukthankar [20] consider the main roles agents can have on human-robot teams: agents support individuals (as personal assistants), agents support teams (as in communication or coordination), and agents act as equal team members. A huge factor is making agents and humans understand each others' intentions and responses. Additionally, human interactions depend on teammate predictability, shared understanding, redirection and adaptation abilities. Making sure that agents have all this pertinent information depends on clear communication and trust in "normal" behavior on other agents' parts.

Previous research has also examined coordination in robots, such as "combined coordination cost measure" (CCC) which increases as group productivity decreases [18]. The CCC allows the authors to measure the cost of coordination, dynamically work to

decrease it, and subsequently increase productivity. Work has also examined the role of non-verbal communication between humans and robots [2] and how it might impact teams. Other relevant research on teamwork, member roles, communication, and human- agent teams (of varying distributions and complexities) includes [8, 21].

We have illustrated the multiple topics inherent in our team of humanitarian robots: modelling the internal mental model, establishing the team structure, trusting other agents, considering how past performance affects an agent, accounting for problems in the environment, and accomplishing tasks despite communication or other failures.

## 3 Problem description

We begin with a high level description of why human teamwork provides valuable insights, move to teamwork in agents and simulators, and then discuss how a specific example of an environmental problem impacts team performance. Consider the real-world example of a wireless communication range, in which a wireless signal can cover a smaller distance than the map size where the agents operate. We model this by implementing a limit on the distance of agent communication, and seek to help the team with imperfect communication perform as fast as a team with perfect communication. We question whether this requires updates to the mental model.

First, we discuss mental models in teams in more detail. A shared mental model[6] explains how each agent reacts to his fellows and environment with communication, action, information gathering, or some other goal-oriented activity. Team coordination and task assignment is made clear through implicit and explicit communication among team members. Teams are devoted to a particular task, which has some life span, and that task dedication brings extra communication and process overhead (from creating, organizing, and disbanding the team) along with the gain in efficiency.

While research indicates shared mental models are important to understanding how a team can communicate and operate to help members, there are few formal measurements of exactly how "shared" these shared mental models are and how helpful the model is to the task at hand. Work by Jonker et al. [12] seeks to solve this by formalizing and implementing a measure of the sharedness of the team mental model and relating this sharing to team performance. Besides the accepted mental model components of technology, task, and team interaction abilities needed by teams [15], the authors introduce new component models of the overall mental model: domain, competence/capabilities, and organization. They also include a "relevance relation" which links components to impact on team performance; the relation depends on circumstances, team task, domain, and performance criteria.

Knowledge about the task, the tools used, and how to reason and interact with teammates is a great start, but Jonker et. al. reason that additional world and agent information will assist the team goal. More knowledge about the world domain at hand, as well as about other agents' intentions, will assist developing the team mental model and give us another way to measure team effectiveness. To demonstrate the formalizations, the authors used the BlocksWorld for Teams (BW4T) application, implementing three-agent teams with varying models but the same measure of task performance. The testbed is explained in detail in Section 4.1.

Our research approaches the mental model from the angle of world problems. We want to inform the mental model structure using analysis of common team problems. Thus, our research addresses the following questions:

- How do environmental factors, like limited communication range, impact the team?
- How can we improve team performance within these constraints?
- What is the role and the impact of having a shared team mental model?

To answer these questions, we limit our study to a specific environment and set of tasks, limit the range of communication for the agents, design and implement new helper agents, and analyze the results.

## 4 Experimental Work

Our experiments were conducted in the BlocksWorld for Teams (BW4T) environment, version 2.0, published by Delft University of Technology. The BW4T environment can be used in the GOAL IDE, using the Prolog language, or through the Repast environment, using the Java language. This work uses the Java language.

### 4.1 Testbed

BlocksWorld is a classic planning domain in artificial intelligence – an agent is tasked with stacking blocks into a vertical stack, moving only one block at a time, and without moving a block on the bottom of the stack. BlocksWorld for Teams (BW4T) from Delft University of Technology [4] adds complexity to the problem by expanding the environment from a table to a map with multiple rooms, limiting visibility of where the blocks are, and allowing the user to have any number of robot or human players on a map. The map contains rooms, hallways, zero or more blocks of various colors inside a room, and a number of agents.

The agent team's goal is to deliver a sequence of blocks in the assigned order. To accomplish the goal, agents have to explore the rooms, find the blocks, and deliver them in the required order to the DropZone. Boxes that are dropped outside rooms and outside the DropZone disappear.

BW4T allows up to one agent in a room at a time, any number of agents in a hallway, and any goal sequence of blocks for the agents to find and deposit in the DropZone on the south end of the map. Agents cannot see other agents, but they can hear other agents' messages, and they can see when a room is occupied.

Fig. 1 shows nine rooms, RoomC1 . . . RoomC3, RoomB1 . . . RoomB3, and RoomA1 . . . RoomA3, as well as the DropZone. Underneath the DropZone, the user sees the required sequence of blocks to be delivered, as well as which blocks have been delivered. In this case the sequence is blue, white, red, cyan, orange, and magenta; the blue block has been delivered (indicated by a green triangle). Agent Smith has picked up a white block in RoomC3, indicated by the agent's change in color from black to white.

An agent cannot see blocks until it moves inside of the room, as shown in the cross section of the environment in Fig. 2. If we assume the agents have memory, they can remember in their own mental model what blocks they have seen and where. This will

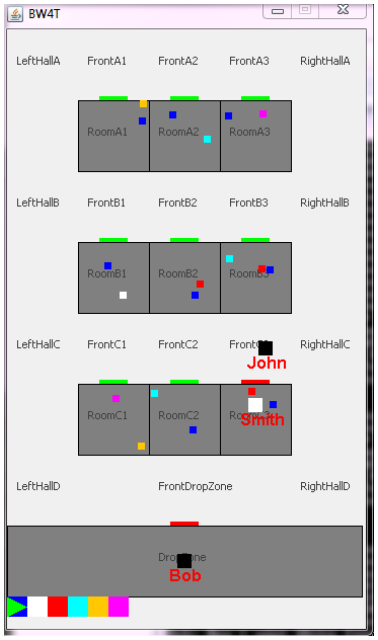


**Fig. 1.** BlocksWorld For Teams (BW4T)

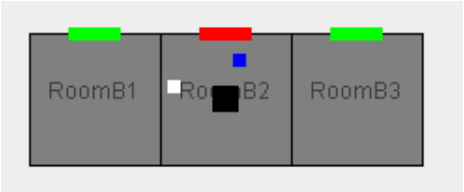


**Fig. 2.** An agent (black) in a room; it can view only the objects (a white box and a blue box) in this room.

enable them to retrieve blocks of specific colors they have seen without having to repeat the exploration phase. Obviously when multiple agents run around looking for blocks and do not share all information they have about the blocks found, memory has a limited value. Another agent could go into a room and pick up a block, rendering the information stored in an agent's memory out of date.

Agents are able to communicate with each other through predefined questions, answers, and statements, such as "Who is in RoomB2?", "I am in RoomB2," "Someone, we need a Blue block," and "Ok." [4]. Humans can participate in the BW4T world by controlling one agent, which has the same environmental constraints as any other agent; the human-controlled agent will hear and say messages the same way.

Our agents use a control process similar to the agent decision model from [12], shown in Fig. 3. The main differences discernible at such a high level are that first, after failing to find a block color, our agents continue to search for that color, in case a different agent had been holding that block and has now deposited it in a room. The original decision cycle told an agent to stop searching for a block color after failing to find it, assume that the color was found, and update its goal sequence. Second, our agents save blocks in the room closest to the DropZone instead of any nearby room.

An agent starts without knowing where any blocks are, so it must choose a room to look in for the next color block. If it finds it, it can pick up the block and carry it to some location (either the DropZone to deposit, or to a nearby room for storage). Then the agent decides again – is the task finished? Should it explore another room? Does it

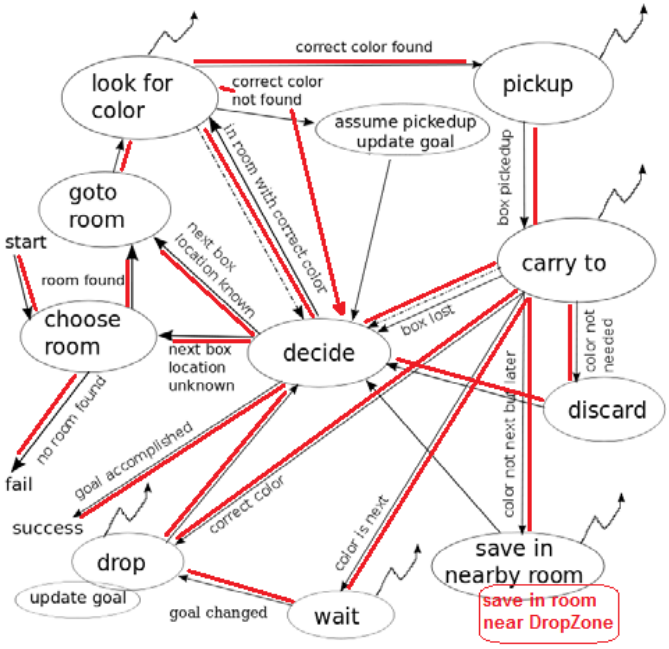


**Fig. 3.** Abstract decision cycle for regular agents in BW4T. [12] model is in black (thinner lines, ovals surrounding action points), with our model overlaid in red (thicker lines, rounded boxes for differing action points).

know where the next color is? When other agents communicate information, an agent's job is easier because it no longer needs to explore some rooms (if it just heard what blocks are in a particular room) or because it no longer needs to get every block in the sequence (if another agent has just informed the team it will pick up the next block).

### 4.2 Approach

The first task was to implement a scalable team model that was robust to failures, remembered required task information, and could communicate without overburdening other agents with communication processing. The second task was to implement the limited communication range for the agent team, design and implement useful helper agents, and record the resulting task performance.

Our hypothesis was that with the addition of communication helpers, our team should be able to coordinate better and improve (decrease) the time taken to finish the task.

While developing the agents and the model, we kept the following key topics in mind: implication of interdependence of team members [16]; agent communication; beliefs, desires, intentions [6]; what factors affect task performance [20]; technology used; how to share job or task models; and how the team interacts [15]. While some items were easily known (such as the task model: get all colors in the color sequence

into the DropZone), we are most interested in how the team members interact. We can choose which messages agents should say or listen to, and we need to account for repeated information and processing power required for message interpretation. We must also choose what domain information (from the world or the agent's intentions) an agent should communicate and remember from others' messages.

To make the team as capable as possible, we built a team without leaders; the first agent to communicate an intention 'wins' that portion of the task. However, we also wanted to account for situations where agents fail, as real agents might in a physical environment due to unavoidable environmental accidents or agent hardware failure. Therefore, while agents will predict future needs in the form of which block to fetch next based on communicated intentions, the agents limit the lookahead task and at one point will revert to fetching the next block required in the sequence.

**How the team interacts:** Agents are responsible for
- speaking and hearing messages from other agents;
- remembering what they have been told (agents do not need to confirm or perform handshakes to verify information was received);
- sending messages when they have changed the environment (i.e., picked up a block or dropped one off);
- sending messages when they have detected something in the environment (i.e., the agent saw a block in a room);
- making sure duplicate messages are handled (other agents may communicate the same information several times).

**World domain information that agents should communicate:** block colors in a room; room names; deposited blocks in the DropZone; which blocks are removed from a room.

**Agent domain information that agents should communicate:** intention to pick up a block; intention to go to a room; which room the agent is in; when an agent picks up a block and what color it is.

### 4.3 Experimental setup

We started with a baseline of task completion times for a team of three agents. The team had perfect communication: all agents could hear all other agents regardless of location, all agents were regular-type, and all agents communicated all types of information. This baseline is given in all results graphs as a black short-dashed line with the label "Baseline – perfect."

Our goal was to develop a team of four (three regular agents, and one helper agent) with limited communication that performed as well as the team of three with perfect communication. Previous research [12] explored how communicating different types of information (none, world, intention, or all) assists agents in task completion, so all of our agents communicated or repeated both information types. World information includes facts like "There is a Blue block in RoomA1," while intention information includes goals like "I will get an Orange block from RoomB3."

We modified the BW4T environment by limiting communication range to a radius roughly the size of one, two, or three rooms. Agents can hear messages originating from within the circle of this radius centered on them.

Our experimental results surprised us. An exploratory agent helped only when the communication limitation was neither very small nor close to no communication limitations. very large.

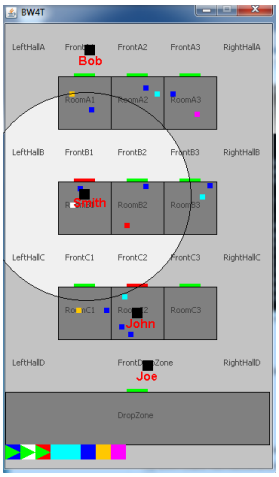


**Fig. 4.** Agent Smith can hear any message originating from within the circle around it. Currently, all other agents are out of range.

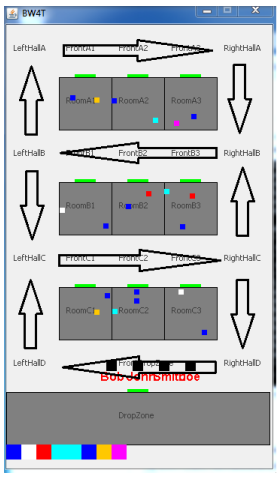


**Fig. 5.** The Figure 8 agent route. The agent repeats the route until the task is completed.

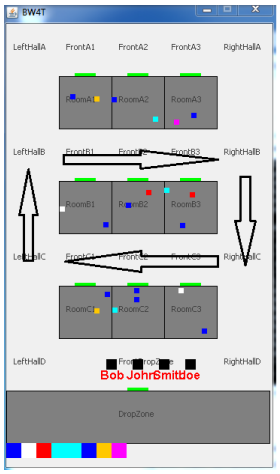


**Fig. 6.** The Circler agent route. The agent repeats the route until the task is completed.

We next considered facets of the environment that helper agents might take advantage of. Because the environment is small and may have widely distributed goals (blocks in far-away rooms), we first designed an agent that would walk the entire map. Next, we noted that even with a reduced radius, most agents would be able to hear communications coming from the center of the map. We designed two agents to take advantage of this – a traveling agent and a stationary agent. All three agents were designed to help spread communications from one end of the map to the other. Lastly, we created an agent for pure reconnaissance, which also repeated messages as an afterthought rather than as its primary purpose. The four additional agent types we created are:

- Figure 8: the agent follows a figure 8 pattern on the map (see Fig. 5).
- Circler: the agent makes a large circle in the middle of the map, around the B rooms (see Fig. 6).
- Scout: the agent explores all rooms, communicating information about the room contents but not attempting to deliver any blocks. The order of the visits is room C1 to C3, B3 to B1, A1 to A3, and random after that sequence is complete.
- Town-crier: the agent posts himself in one location (FrontB2) for the duration of the task.

In the first three cases wherein the agent moves, it repeats the last message heard when it reaches an intersection. This helps reduce interpretation and computation overload on other agents. For simplicity's sake, the communicator filters out "I am going to RoomX" messages. When agents start in the environment, they are so close that they are able to hear each other regardless of communication radius. We thus stop the communicator from repeating this message type, to prevent initial confusion (e.g., Agent A says it is going to RoomA1, the communicator repeats this message, and Agent A decides that since another agent has said it will go to the same room, Agent A will update its next destination and go to a different room). The town-crier repeats messages every one-half second. Each team consisted of three agents who attempted to deliver all blocks in the proper order, with one extra teammate of exactly one of the four new communicator agent types. In all cases, the regular agent types knew how many regular agent types existed in the environment, but not how many communicators; regular agents did not try to take advantage of the helper in any way other than to repeat messages often in the hope that other agents will receive those messages. The regular agent, in turn, cannot assume that other agents really received its messages, and may repeat the last message sent even if all agents were able to hear the message. We took measurements of task completion (using each block delivery as a marker) for each team type for each radius value: 10 (one room width), 20 (two rooms width) and 30 (three rooms width).

### 4.4 Results

As mentioned previously, each team was made of three "regular" agents (each tries to deliver blocks) and one "communicator" agent. We give a baseline (the solid black line in each results graph) for each communication distance. The baseline is built from a team of three agents who communicate with the specified radius, and a fourth teammate who wanders through hallways communicating and contributing nothing.

We tested against four different environments, to reduce the chance of biasing any particular agent type, as seen in Fig. 7. Two maps were heavily biased in block location – either the top two or bottom two rows in the map. In two maps blocks were randomly located without any empty rooms. Each agent team with different helper agents was tested in each map and the results averaged per team.

Results for team helper type for three different communication ranges on a time vs. block delivery graph are given in Figures 8, 9, and 10. Each graph contains two baselines: the average time taken by a team with perfect communication to do the task, and the average time taken by a team of three regular agents with the specified range.

Recall that our goal is to help limited communication teams perform faster; therefore, the closer a team's average block delivery times are to the perfect communication team's average, the better our team has done. In the graphs, we see a decrease in average block delivery time as increased team performance. We also consider any team that performs better than the limited range baseline team to be a success, even if it did not match the perfect team's performance.

In all cases, we expected at least a very small increase in performance for all agent communicator types. We expected communicator agents to repeat necessary and useful information (block locations) at some point during their travels (though they are also likely to repeat intentions or traveling-to messages), alerting nearby agents to room

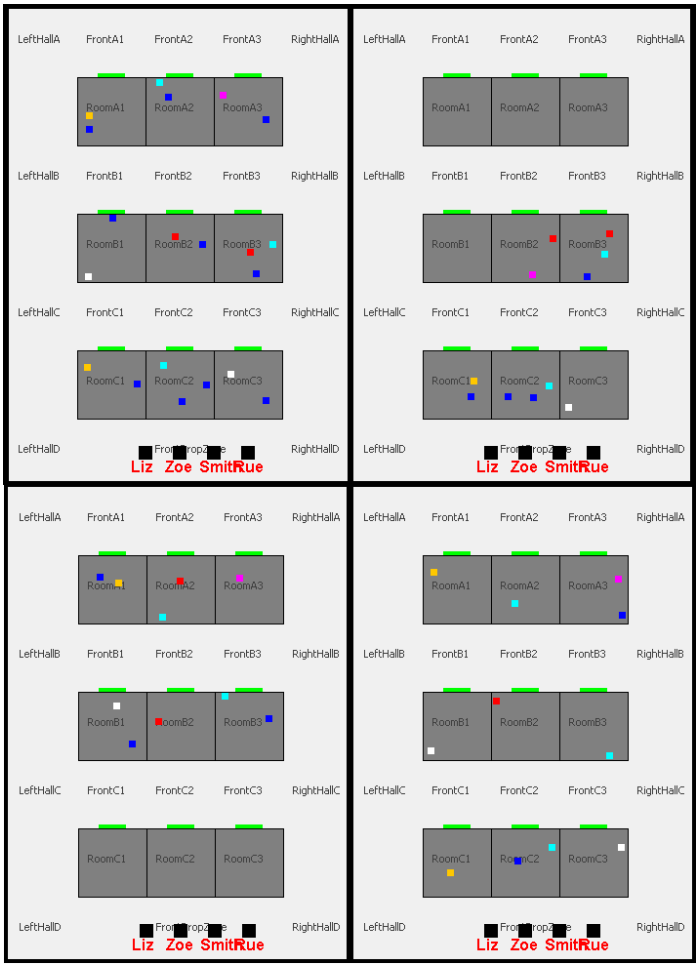


**Fig. 7.** The four maps used in the experiments.

contents. Because agents always are searching for their next chosen block, the communicators will give regular agents new information as to which rooms contain which blocks, in cases where agents had not heard the original communication. Thus agents who were blindly searching will come to know which room they should go to next.

In teams of agents working with a reduced communication range of 10 units (about 1 room width), no agent helper improved results up to the perfect baseline; furthermore, the added overhead actually hurt the teams. The team with the Circler agent was the only one to come close to the baseline of regular agents with a radius of 10. We did not expect agent teams with reduced communication range with helper agents to perform worse than teams with reduced communication range only, but additional agents may carry stale data throughout the environment and give fellow agents the wrong impression about the current world state. For example, a helper agent may repeat information

about Room A1 and the blocks contained within, causing a regular agent to 'think' the room contains a block that is no longer in it. The regular agent may update its state information about the room every time it hears the agent, because the helper agents do not try to keep track of stale data. A helper agent with incorrect information doesn't try to verify the information; additionally, the designed helper agents do not interact with other agents to see which other agents have the most up to date information. A way to improve and counteract this misinformation would be to keep a time stamp of information (in this environment, the most recently received message about a room's contents).

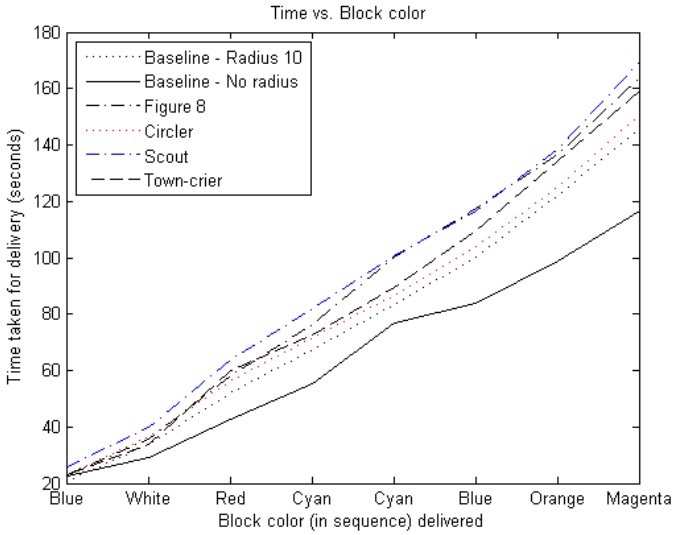


**Fig. 8.** Average delivery time for each block with a communication range of 10.

For teams of agents working with a reduced communication range of 20 units (about 2 room widths), only the addition of the Scout agent actually assisted the team; teams with Figure 8 and Circler agents did no better than the team with no helper agent. The team with the Town-Crier agent did significantly worse, possibly because it communicates more often to team members than other helper agent types and therefore passes more stale information, or because it communicates too often and distracts team mates (adds overhead without adding value).

The useful agent, the Scout, halved the time difference in performance between the agent team without reduced communication and the team with reduced communication.

Finally, the teams of agents working with a reduced communication range of 30 units (about three room widths) were not assisted by any helper agents. An interesting note is that while neither agents with a very small communication range (of 10 units) or a larger communication range (30) units were helped, the team with the larger range was impacted more negatively than the other teams.

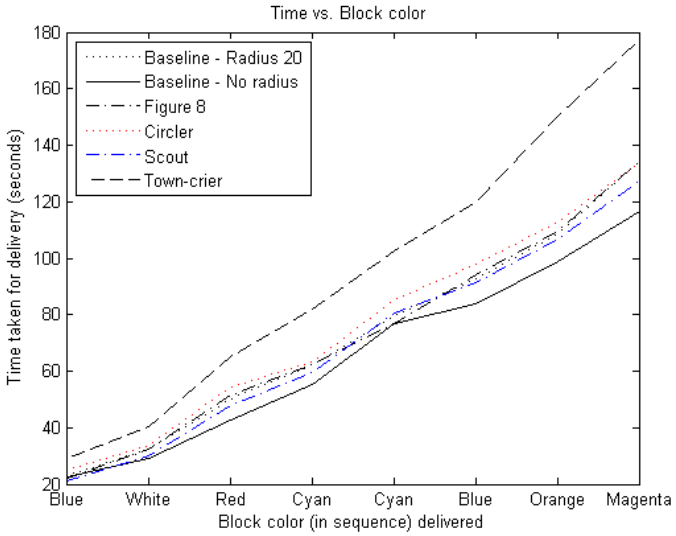


**Fig. 9.** Average delivery time for each block with a communication range of 20.

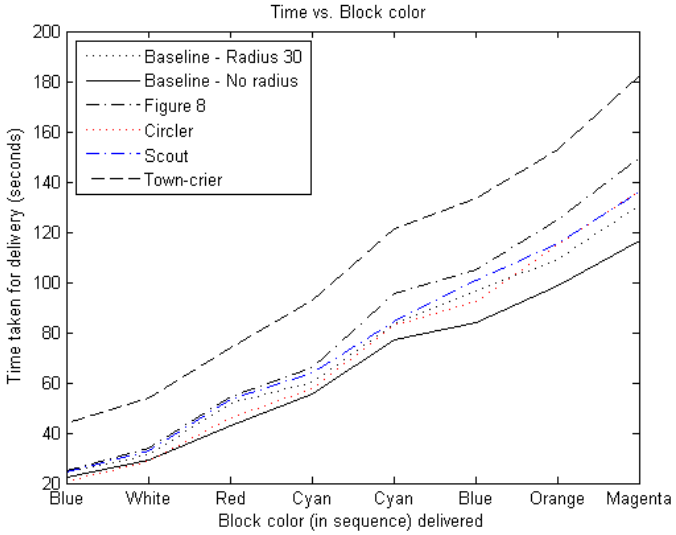


**Fig. 10.** Average delivery time for each block with a communication range of 30.

From these results we can draw a few conclusions: exploratory agents (with some added communication ability) were of more help than simple repeaters of information; adding helper agents can be either detrimental or beneficial; and that helper agents may be of more assistance when the team's communication range is neither very small

nor very large compared to the environment. In either of those two cases, the added overhead may simply not be worth the time and expense of adding assistance.

## 5 Future Work

In future work, we will expand the environment size, investigate interesting performance results in which teams performed much differently than expected (better or worse), and model different sorts of communication failures.

Making the environment larger, say into a $4 \times 3$ or a $5 \times 5$ room set, will help indicate if the results found in the smaller room setup are scalable and if the same environment features can inform agent type. For example, if a $5 \times 5$ room set has hallways mixed into the rows of rooms (say between rooms A2 and A3, B3 and B4, C2 and C3, etc.), an agent similar to the Figure 8 agent will now take much longer to reach the original point. We will also investigate the simulator's potential in adding multiple-exit rooms, dead-ends, and other features that make navigation more difficult or complicated for agents.

We would like to explore and verify causes of poorer performance in teams for specific additional agent types, and we will also examine unexpectedly better performance by certain agents. For example, is our theory on what communication feature makes the Figure 8 agent decrease performance correct? We will also explore what might possibly make teams of three agents with extra agents perform better than our perfect baseline, as in the case of a Figure 8 agent team with a radius of 30. Along these lines, we will make several improvements or options on our existing communicator agents:

- repeat world information (facts) only, such as room contents, instead of agent intentions as well as facts
- repeat room contents as the communicator passes that room
- use a priority to decide which messages to repeat

We intend to explore how repeating different information types (world facts vs. agent intentions) affects the team. For example, we suspect that allowing communicator agents to repeat intention information may hinder some teams instead of improving performance. In some cases, it might save processing time and therefore speed-up agent task assessment if the communication does not repeat intentions. Additionally, implementing responses (such as "ok" to a received message) may drastically cut down on the number of messages sent and received. We will also develop at least one additional agent type – a Follower, who moves to the location from which its last message was heard. This should allow the Follower to trail other agents and serve as a sort of echo for previous communications.

We will implement message loss in the form of some k% (for example 10% or 20%) of messages getting completely dropped. This simulates environments with bad reception, like dropped cellular calls or environments with a lot of radio interference.

A more complex and interesting future direction we would like to take is a team vs. team challenge. Because the BW4T environment supports both humans and agents, as well as a server-client setup, we would like to include one to two humans with two to three agents in a race to find a sequence of blocks. This might be a programming

challenge (each team is allowed to code their robotic agent) or a simple usage challenge, where the human team member must learn how best to interact with the robotic agent. Lastly, we are also interested in additional metrics, such as how much work duplication (efficiency loss) occurs because of lost communication.

These experiments suggest updates to the structure of the team mental model itself:

- Can we update the organizational model with environment features, such as expected or known flaws like communication limitations? For example, if the environment will limit range communication range, can we guess a nice ratio of communicator-type agents to regular-type agents to improve team performance? Some communicator agents might also serve double duty by exploring while they are on the way to pass messages.
- Are there cases where agents should swap roles? For example, due to a regular agent's failure, an agent switches from messenger to regular agent. We may want some minimum number of regular agents operable at all times, and convert communicator agents when a threshold is reached.
- Are there certain time periods during task execution when the number of a particular agent type should increase or decrease? For example, at the beginning of the task agents should have more communicator agents, but in the middle of the task execution, it might be more efficient to have all agents performing the task without extra communicators.
- Can we predict the most helpful agent types based on simulated environments? For example, are agents that focus on reconnaissance always more helpful than agents who repeat messages in environments where exploration is required?

Answering these questions can help us include environment features in team models, and improve completion rate of a task before we test in the real environment.

Another piece of future work is implementing and extending the concept of the Gatekeeper [1, 22]. The Gatekeeper agent is responsible for collecting information about agents in the environment, deciding if some set of those agents can accomplish a task, and assigning those agents to a team to do that task.

The Gatekeeper can further be improved by using a Monitor agent to monitor the status of the team and the team members (for hardware failure or other unexpected occurrences). The Gatekeeper, having knowledge of the tasks and task requirements throughout the environment, may need to move around the environment and assign one or several teams in one or several different tasks.

The Monitor agent would be responsible for notifying the Gatekeeper agent of sudden team failures. Why have a Monitor agent at all? Say that the agents assigned to a team by the Gatekeeper have cheap, local communication abilities. If the Gatekeeper moves far enough away from the team, each team member may no longer reach the Gatekeeper with messages. This means the team either needs to be permanently equipped with more robust communication requirements (in case failures occur which require notifying the Gatekeeper), or the team needs to send a team member to the Gatekeeper to alert it of team failure (allowing the Gatekeeper to reassign task roles, etc). The first option causes more expense than necessary; after all, why equip 20 agents scattered throughout an environment with expensive, robust equipment that they might

not need? The second option removes a team member for some unknown length of time from action in the environment – potentially negatively impacting the team. Assigning the Monitor agent to collect failure results and reporting them to the Gatekeeper allows us to equip fewer agents with expensive hardware as well as reducing the handshaking or other protocol overhead of discussion with the Gatekeeper agent.

Further work with agent helpers will integrate Gatekeeper and Monitor agents for heterogeneous agents and tasks within the BW4T environment. In this way, we seek to further improve agent teamwork with reduced agent capabilities and distributed decision making.

## 6 Conclusions

We motivated our research with an example of a humanitarian team of robots, and reviewed a range of research impacting the team. Using this example, we chose a manageable subproblem of limited communication range and implemented helper agents to adjust our team's performance. We generated results using the multi-agent system simulator BlocksWorld for Teams (BW4T), and showed that teams can be assisted by new agent types to accommodate for the environment problem. We showed that communicator agents can improve or decrease the rate of task completion when compared to a team without the extra communicator. Lastly, we proposed additional considerations for mental models, and gave future research for communication and teams with the BW4T simulator.

**Acknowledgments.** The authors would like to thank Birna van Riemsdijk for her much needed assistance in the BW4T setup, and her intrepid students and creators of BartJohnAgent.java, which contributed much knowledge and example environment function calls.

This material is based upon work supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. NSF/IIS-1208413, NSF/IIS-1216287, and NSF/IIS-1216361. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

## References

1. Huib Aldewereld, Virginia Dignum, Catholijn Jonker, and M. van Riemsdijk. Agreeing on role adoption in open organisations. *KI - Knstliche Intelligenz*, 26:37–45, 2012. 10.1007/s13218-011-0152-5.
2. C. Breazeal, C.D. Kidd, A.L. Thomaz, G. Hoffman, and M. Berlin. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 708 – 713, aug. 2005.
3. Gentner Dedre and Albert L. Stevens, editors. *Mental models*. Lawrence Erlbaum Associates, London, 1983.
4. Delft University of Technology. *BW4T2 Specification*, 2011.
5. J. Alberto Espinosa, Robert E. Kraut, Sandra A. Slaughter, Javier F. Lerch, James D. Herbsle, and A. Mockus. Shared mental models, familiarity and coordination: A multi-method study of distributed software teams. In *Int'l Conf. in Information Systems*, pages 425–433, 2002.

6. Xiaocong Fan and John Yen. Modeling and simulating human teamwork behaviors using intelligent agents. *Physics of Life Reviews*, pages 173 – 201, 2004.
7. S. M. Fiore, E. Salas, H. M. Cuevas, and C. A. Bowers. Distributed coordination space: toward a theory of distributed team process and performance. *Theoretical Issues in Ergonomics Science*, 4:340–364, 2003.
8. Joseph A. Giampapa and Katia Sycara. Conversational case-based planning for agent team coordination. *Case-Based Reasoning Research and Development; Lecture Notes in Computer Science*, 2080/2001:189–203, 2001.
9. Andrew Howard, Lynne E. Parker, and Gaurav S. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *The International Journal of Robotics Research*, 25(5-6):431–447, 2006.
10. Matthew Johnson, Jeffrey M. Bradshaw, Paul J. Feltovich, Robert R. Hoffman, Catholijn Jonker, Birna van Riemsdijk, and Maarten Sierhuis. Beyond cooperative robotics: The central role of interdependence in coactive design. *IEEE Intelligent Systems*, 26(3):81–88, May/June 2011.
11. Philip N. Johnson-Laird. *Mental models*. Harvard University Press, Cambridge, 1983.
12. Catholijn M. Jonker, Birna van Riemsdijk, Iris C. van de Kieft, and Maria Gini. Towards measuring sharedness of team mental models by compositional means. In *Proc. 25th Int'l Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)*, pages 242–251, 2012.
13. Catholijn M. Jonker, M. Birna van Riemsdijk, and Bas Vermeulen. Shared mental models: A conceptual analysis. In *Workshop on Coordination, Organization, Institutions and Norms in Multi-Agent Systems at AAMAS*, pages 132–151, 2010.
14. Richard Klimoski and Susan Mohammed. Team mental model: Construct or metaphor? *Journal of Management*, 20:403, 1994.
15. John E. Mathieu, Tonia S. Heffner, Eduardo Salas, and Janis A. Cannon-Bowers. The influence of shared mental models on team process and performance. *Journal of Applied Psychology*, Vol. 85, No. 2:273–283, 2000.
16. Carol R. Paris, Eduardo Salas, and Janis A. Cannon-Bowers. Teamwork in multi-person systems: a review and analysis. *Ergonomics*, 43, 8:1052–1075, 2000.
17. I Rekleitis, V. Lee-Shue, Ai Peng New, and H Choset. Limited communication, multi-robot team based coverage. In *Proc. Int'l Conf. on Robotics and Automation*, volume 4, pages 3462 – 3468, 2004.
18. Avi Rosenfeld, Gal A. Kaminka, Sarit Kraus, and Onn Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6–7):633–655, 2008.
19. Paul Salmon, Neville Stanton, Robert Houghton, Laura Rafferty, Guy Walker, Daniel Jenkins, and Linda Wells. Developing guidelines for distributed teamwork: Review of literature and the HFI DTC's distributed teamwork studies. *HFIDTC*, 2008.
20. Katia Sycara and Gita Sukthankar. Literature review of teamwork models. Technical Report CMU-RI-TR-06-50, Robotics Institute, Carnegie Mellon University, November 2006.
21. Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, pages 83–124, 1997.
22. M.B. van Riemsdijk, V. Dignum, C.M. Jonker, and H. Aldewereld. Programming role enactment through reflection. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 2, pages 133 –140, aug. 2011.
23. A. Wagner and R. Arkin. Multi-robot communication-sensitive reconnaissance. In *Proc. Int'l Conf. on Robotics and Automation*, volume 5, pages 4674 – 4681, April-1 May 2004.
24. Michael A. West, Dean Tjosvold, and Ken G. Smith, editors. *International Handbook of Organizational Teamwork and Cooperative Working*. Wiley, 2003.