On Proactive Helping Behaviors In Teamwork

Sen Cao, Richard A. Volz, Thomas R. Ioerger, and Michael S. Miller
Department of Computer Science, Texas A&M University
College Station, TX 77843, USA
{sencao, volz, ioerger, mmiller}@cs.tamu.edu

To Dick Volz

for his unique combination of sharp insight, quick wit and patient guidance, which made my research endeavor full of challenges and fun, and made me understand that teamwork modeling may not be just a mechanism for agents, but also a way of human lives.

-Sen Cao

***International Conference on Artificial Intelligence* (IC-AI). Las Vegas, 2005.**

On Proactive Helping Behaviors In Teamwork

Sen Cao, Richard A. Volz, Thomas R. Ioerger, and Michael S. Miller
Department of Computer Science, Texas A&M University
College Station, TX 77843, USA
{sencao, volz, ioerger, mmiller}@cs.tamu.edu

Abstract - Teamwork has become increasingly important in diverse disciplines. Cognitive studies on teamwork have shown that team members in an effective team often have mutual expectations based on their shared mental models and proactively offer assistance to each other. We present a formal model called Role-Based Proactive Helping Behaviors (RoB-PHB) to enable proactive assistance among (sub)teams. Through RoB-PHB, agents can dynamically identify others' help needs and provide helps by a course of actions on the fly. We have designed algorithms to implement our RoB-PHB formalism in a teamwork architecture called Role-Based Collaborative Agents for Simulating Teamwork (RoB-CAST). Our experiments on RoB-CAST have shown that the team with proactive helping behavior achieved better team performance.

Keywords: Teamwork, Role-Based Shared Mental Model, Helping Behavior

1 Introduction

In recent years, teamwork has become increasingly important in diverse disciplines, from business management to sports and entertainments to defense simulations and to virtual training. In dynamic and distributed environments, teamwork is more than an aggregation of coordinated individual actions. Several teamwork models have been developed to explore the critical underlying mental states that drive agents to perform their individual actions while leading to a team effort, such as joint intentions [5], SharedPlan theory [6], and joint responsibility [7, 8]. Based on these teamwork models, various teamwork architectures have been designed to support agent teamwork, such as the BDI architecture [15], STEAM [17], GRATE [9], and CAST [18].

Moreover, cognitive studies on teamwork have shown that team members in an effective team often maintain shared mental models [3, 12]. Cannon-Bowers et al. suggested that shared mental models are "knowledge structure held by members of a team that enable them to form accurate explanations and expectations for the task, and in turn, to coordinate their actions and adapt their behavior to demands of the task and other team members" [3]. Klimoski and Mohammed insisted that "there can be (and probably would be) multiple mental models co-existing among team members at a given point in time" [10]. Shared mental models could contain task-specific knowledge, task-related knowledge, knowledge of teammates and attitudes/beliefs and "shared" means overlapping, similar, identical, complimentary and/or distributed [4]. Empowered by shared mental models, agents in an effective team bear mutual expectations to each other; in particular, they can dynamically identify others' help needs and proactively offer assistances on the fly.

Brehm and Kassin identified the motivations of helping behavior from biological factors (i.e., a creature has a tendency, originated from natural selection, to reciprocal helping), emotional factors (i.e., a person's empathy makes him/her help to reduce the distress of another person) and social normative factors (i.e., social norms promote help giving in social contexts) [1]. Lind proposed a dual-aspect-theory of moral development and helping behavior to distinguish a person's desire to help and his/her ability to help adequately and further hypothesized the conditions of triggering helping behavior [11]. The dual-aspect-theory of helping behavior can be viewed as a hypothesis about how social norms actually motivate helping behavior. Miceli, Cesta and Rizzo described the conditions and motivations for seeking and giving help based on social dependence between two agents (with attitudes of help seeking and giving respectively) [13].

Backing up behaviors are a special kind of helping behaviors in the situation where some team member(s) fails to accomplish a certain action and other team member(s) take an action to cover what the failure action targets. Porter et al. proposed a Five Factor Model of personality to describe the key characteristics of backing up behaviors, including back up recipients, back up providers, and the legitimacy of the needs for backing up [14].

In this paper, we present a formal model of proactive helping behavior based on shared mental models. While many social morals, such as laws, religions and cultures, might affect the decisions on helping behaviors, we focus on how to identify help needs and how to provide helping behaviors correspondingly. Unlike social dependence that only enables helping behaviors between two agents, our model facilitates proactive helping behaviors among teams.

Our model enables two types of helping behaviors: 1) taking over what others are doing if they fail (called backup behaviors), and 2) helping others to achieve conditions required by what they are doing (called promotion behaviors). Based on shared mental models, agents can identify help needs for these types of helping behaviors and initialize courses of actions to meet the needs if they can.

In next sections, we will first briefly describe our teamwork architecture called Role-Based Collaborative Agents for Simulating Teamwork (RoB-CAST), including the specification of teamwork knowledge and Role-Based Shared Mental Models (RoB-SMMs). Then we will explain our formal model of Role-Based Proactive Helping Behaviors (RoB-PHB) to facilitate the above two types of helping behaviors among teams. We will present the algorithms that implement RoB-PHB in our RoB-CAST. We will also describe our experiments to show that our RoB-PHB enables agents to achieve better team performance. Finally, we will summarize the contribution of this work and discuss further improvements.

2 Overview of RoB-CAST

RoB-CAST has been developed to simulate effective teamwork based on shared mental models. Teamwork knowledge is specified by a teamwork programming language called Role-Based Multi-Agent Logic Language for Encoding Teamwork (RoB-MALLET). Agents in RoB-CAST maintain Role-Based Shared Mental Models (RoB-SMMs) and represent the teamwork knowledge in their RoB-SMMs. In particular, agents maintain task-specific knowledge by team processes in RoB-SMMs. Through team processes, agents coordinate with each other for the execution of teamwork; moreover, agents can be mutually aware of what others are doing, and further activate various reasoning mechanisms to improve team performance, such as proactive helping behaviors discussed in this paper.

In this section, we will briefly describe RoB-MALLET, and how task-specific knowledge is specified therein. Then we briefly describe RoB-SMMs, particularly how they represent task-specific knowledge as team and how agents maintain team processes during the execution of teamwork.

2.1 RoB-MALLET

RoB-MALLET has rich expressivity for teamwork knowledge. It contains a variety of constructs for specifying operators, plans, team structures, shared goals, and team processes. From the perspective of the task-specific knowledge, RoB-MALLET is distinguished from other agent/team programming language in two aspects: 1) RoB-MALLET specifies team processes explicitly by using the mental states underlying teamwork, as discussed in existing teamwork models [5, 6, 7], such as, mutual beliefs, shared goals, and joint intentions; 2) the specifications of team processes are in terms of conceptual notions (roles and role variables) instead of specific agents, allowing reuse by different teams of agents.

The primitive actions in team process are operators that are executed by agents in the domain. RoB-MALLET specifies an operator by a set of preconditions and a set of effects. The execution of an operator transits the domain from a state in which its preconditions are satisfied to another in which its effects are satisfied. Before executing an operator, an agent evaluates the preconditions based on its individual beliefs and asserts the effects of the operator in its individual beliefs after the execution of the operator. There are three modes of handing false preconditions: *fail*, *wait* and *achieve*. Suppose the preconditions of an operator *op* are false. In a *fail* mode, an agent does not execute *op* but continues with the next action[1]. In a *wait* mode, if the preconditions become true within a specified period of time, *op* is executed. Otherwise the behavior is as in the *fail* mode. In an *achieve* mode, an agent tries to achieve the preconditions. If the preconditions are achieved, the agent executes op; otherwise the agent does not execute *op* but continues with the next action. E.g., an operator for moving to a square (?x, ?y) in a wumpus world is specified as (ioper movein (?x ?y) (pre-cond (not (wumpus ?x ?y)))).

In RoB-MALLET, a team process is specified by a role-based plan. Similar to an operator, a plan has a set of preconditions and a set of effects and there are the same modes for handling false preconditions. Their semantics are the same as those in an operator, except that plan preconditions and effects are in terms of the performers' mutual beliefs while those of an

[1] A choice construct is also defined in which when a plan or operator fails, an alternative is attempted.

operator are in terms of the performer's individual belief. A plan also has a virtual team of roles, a set of constraints, a set of termination conditions and a process. We require that every action in a team process must be associated with a role (or role variable) or a set of roles (or role variables) , e.g., (Do r1 op1). A role variable is a reference to a role dynamically selected from a list of roles according to concrete situations. An agent is delegated to every role, and must execute the actions associated with the role and those associated with a role variable, which the role is selected to fill. The virtual team consists of all roles in the team process. The set of constraints is a conjunction of literals and specifies the conditions that must be satisfied when delegating the roles in the virtual team to the agents invoking the plan[2], which may involve communication to access others' beliefs.

A process consists of actions. Do constructs associate actions (e.g., operators or plans) with roles and/or role variables, e.g., (Do r1 (movein 3 4)). So, Do constructs specify individual/joint intentions of the agent(s) to which a role is (are) delegated. A variety of constructs can be used to express the flow of the actions, such as sequential, parallel, selection and iteration. The conditions in selection and iteration constructs are evaluated based on the mutual beliefs of individuals involved in the actions controlled by the constructs. Termination conditions are a set of literals, and are used to monitor the execution of the plan, similar to Jennings' conventions [7], If a termination condition becomes true, the plan execution is terminated.

In RoB-CAST, a team of agents starts a task by invoking a plan. Once the plan is invoked, the roles are dynamically delegated to the agents and each agent executes the actions associated with the role(s) and/or role variable(s) delegated to the agent. An agent could be in multiple teams, and a team could be involved in multiple tasks simultaneously.

2.2 RoB-SMM

Each agent has a RoB-SMM and the union of these models forms a complementary, overlapping and distributed mental model. A RoB-SMM contains teamwork knowledge, including operators, plans, team structures (agents in teams and their capabilities), beliefs, shared goals, and team processes for achieving the shared goals. We focus on how team processes are represented and maintained in RoB-SMMs and how RoB-SMMs enable mutual awareness.

The team processes in a RoB-SMM are represented by a tree structure called a team organization and an execution model called an individual process. In a team organization, each node is a tuple (G, φ, P, S, C), where G is a team of agents, φ is a shared goal of G, P is a plan by which the agents in G use to achieve φ, S is the team structure of G invoking P (i.e., the delegations from roles to agents), and C is the set of nodes corresponding to the sub-goals of φ. When a team of agents invokes a sub-plan to achieve a goal, a corresponding node is added under the root of the team organization. For a hierarchical plan, a team organization represents the hierarchy of tasks in teamwork and each node represents the relationships among agents, shared goals, plans for achieving the goals and team structures for executing the plans.

The individual process of an agent contains only the actions related to the agent, rather than the whole team process. An individual process is expressed in an extended Petri net, called RoB-CAST-PN. For each plan, RoB-CAST generates a RoB-CAST-PN representation for a role or role variable to represent the actions associated with it. Every action associated with a sub-plan in the team process is translated into a special transition attached with preconditions, effects, and termination conditions. When an agent together with other agents invokes a plan, the agent dynamically composes its individual process by expanding the transition corresponding to the plan with the RoB-CAST-PN representation(s) of the role(s) or role variable(s) delegated to the agent. The agent shrinks the RoB-CAST-PN representation(s) to the plan transition after the plan is finished. An important feature of RoB-CAST-PN is that the temporal orders between the actions (even those associated with different individuals) are preserved during the translations and compositions.

Although each agent only maintains the actions related to itself, RoB-CAST agents have mutual awareness of what other agents are doing. Agent ag1 can refer its team organization to see what plans agent ag2 is involved in and what role(s) and/or role variable(s) is (are) delegated to agent ag2. Then, agent ag1 can construct the individual process of agent ag2 and further know the actions in it. Agent ag1 can infer the markings of ag2's individual process in multiple ways, e.g.: 1) the temporal orders and/or coordination between ag1's actions and ag2's actions, 2) observations on ag2's performances of actions.

3 Formal Model of Proactive Helping Behaviors

From the behavioral perspective of teamwork, a team of agents (G_1) may provide helping behaviors (a course of action a) to benefit the goal (φ) of another team of agents (G_2). In general, team G_1 can help team G_2 to reach G_2's goal φ in two ways: 1) G_1 takes over G_2's goal φ (called backup behaviors); and 2) G_1

[2] See [2] for a detailed discussion of our roles vis a vis previous use of roles.

achieves the prerequisite condition of to G_2's goal φ (called promotion behaviors). Our model of Role-Based Proactive Helping Behaviors characterizes these two types of helping behaviors by a meta-predicate Help(G_1, G_2, φ, a), which expresses the condition that agents in team G_1 can help agents in G_2 by executing an action a, as follows:

$$\text{Help}(G_1, G_2, \varphi, a) \equiv \text{MBel}(G_1, \text{Goal}(G_2, \varphi)) \qquad 1$$
$$\wedge\ (\text{MBel}(G_1, \mu) \vee \text{MBel}(G_1, \nu)) \qquad 2$$
$$\wedge\ \text{MBel}(G_1, \text{Capable}(G_1, a)) \qquad 3$$

where,

$$\mu = (\Box\neg\, \text{Done}(G_2, \varphi) \wedge \qquad 4$$
$$(\text{Done}(G_1, a) \rightarrow \varphi)) \qquad 5$$
$$\nu = (\exists\ \psi\ ((\neg\psi \wedge \qquad 6$$
$$(\neg\psi \rightarrow \Box\neg\, \text{Done}(G_2, \varphi))\ \wedge (\psi \rightarrow \neg\Box\neg\, \text{Done}(G_2, \varphi))) \wedge \qquad 7$$
$$(\neg(\exists G_3 \exists a'\ (\text{Do}(G_3, a') \wedge (\text{Done}(G_3, a') \rightarrow \psi)))) \wedge \qquad 8$$
$$(\text{Done}(G_1, a) \rightarrow \psi))) \qquad 9$$

Then, Help(G_1, G_2, φ, a) $\rightarrow$ Do(G_1, a).

In the above formula, symbol $\Box$ is the temporal operator "always"; MBel(G, I) means that the agents in G mutually believe I; Goal(G, g) means that the agents in G have a shared goal g; Done(G, g) means that the agents in G have achieved a shared goal g; Done(G, a) means that the agents in G have executed action a; Do(G, a) means that that the agents in G executes action a; and Capable(G, a) means that team G can perform action a.

Clause 2 in Help(G_1, G_2, φ, a) contains two disjunctive clauses and these two clauses characterize two types of help needs. The first clause specifies a help need for a backup behavior, and the second, a help need for a promotion behavior. For easier understanding, we explain Help(G_1, G_2, φ, a) for these two types separately.

For backup behaviors, the agents in G_1 proactively help the agents in G_2 to achieve a goal φ by executing action a if 1) G_1 mutually believes that G_2 has a shared goal φ (clause 1); 2) G_1 mutually believes that G_2 will never reach the shared goal φ (clause 4); 3) G_1 mutually believes that φ will be true if G_1 executes action a (clause 5); and 4) G_1 mutually believes that G_1 is capable of action a (clause 3).

For promotion behaviors, the agents in G_1 proactively help the agents in G_2 to achieve a goal φ by executing action a if 1) G_1 mutually believes that G_2 has a shared goal φ (clause 1); 2) *G1* mutually believes that there exists a condition ψ and ψ is not true (clause 6); 3) *G1* mutually believes that *G2* will never reach the shared goal φ if ψ is not true and that *G2* might reach the shared goal φ if ψ is true (clause 7); 4) *G1* mutually believes that no action performed by any other team can make ψ true (clause 8); 5) *G1* mutually believes that ψ will be true if *G1* executes action a (clause 9); and 6) G_1 mutually believes that G_1 is capable of action a (clause 3).

Based on RoB-SMMs, the agents in G_1 can evaluate the clauses Help(G_1, G_2, φ, a) as follows:

1. G_1 evaluates clause 1 by checking if there is a node in their team organization representing that G_2 is trying to achieve goal φ.
2. G_1 can find the plan P used by G_2 to try to achieve φ through their team organizations, and the termination conditions of P through the plan knowledge. If the termination conditions become true, G_2 terminates the execution of plan P and thus G_2 cannot achieve φ. In this way, G_1 can evaluate clause 4 and identify help needs for backup behaviors.
3. Based on the mutual awareness enabled by RoB-SMMs, G_1 can check clauses 6 - 8. As described earlier, agents can construct other agents' individual processes and thus know the actions other agents are doing. Through the knowledge of operators and plans, the agents in G_1 know the preconditions and effects of *G2's* actions[3]. If *G1* mutually believes that a precondition of one of G_2's actions is false, then *G1* believes G2 might need help (clauses 6 and 7). However, if the agents also know the effect of any ongoing action will imply the precondition (clause 8), they know they do not need to help.
4. Even though planning algorithms may be applied to decide a course of actions for G_1 to achieve φ (or ψ), agents in RoB-CAST currently just search for a plan in their plan library. A plan is a proper course of actions only if its effects imply φ (or ψ), G_1 mutually believes its preconditions, and G_1 can find a team delegation for the roles in the plan to G_1 that satisfies the constraints of the plan and capability requirements of the roles.

In summary, the meta-predicate Help(G_1, G_2, φ, a) gives the conditions under which proactive helping behaviors take place, as well as the specific agents (G_1) and behaviors (a). Although our model is in terms of RoB-SMMs and there may be other representations of shared mental models, this formal works as long as shared mental models enable mutual awareness on team processes. It is important to note that only partial mutual awareness is needed.

4 Algorithms of Proactive Helping Behaviors

Based on the formal model of proactive helping behaviors (PHBs), we implement proactive helping behaviors by a PHB offline algorithm and a PHB online algorithm. The offline algorithm generates all potential helping needs (*fail* and *wait* preconditions of actions) and all potential coverage of possible helping

[3] As the *achieve* mode implies that the executors of an action would achieve its preconditions by themselves, we treat false preconditions in fail or wait modes as help needs for promotion behaviors.

needs (the effects of actions) in a plan. The online algorithm identifies actual help needs based on RoB-SMMs, the potential help coverage generated by the offline algorithm, and then initializes a proper plan to provide helping behaviors. The offline algorithm is run on plans statically before the plans can be invoked as tasks. The online algorithm is run by each agent in a separate thread during the execution of teamwork.

The offline algorithm functions on the basis of the RoB-CAST-PN representations for the roles and role variables in a plan. The offline algorithm checks every transition representing an action (operator/plan) in the RoB-CAST-PN representations and generates potential help needs and coverage. A potential helping need is represented by a 4-tuple (Predicate, Plan, Needers, Transition). A potential coverage is represented by a 2-tuple (Predicate, Plan). The following is the offline algorithm generating potential help needs and coverage for plan P:

```
PHB_Offline(P)
  Create a helping need list HelpNeeds;
  Create a coverage list HelpEffects;
  For each role or role variable r in P, do
     Generate a Rob-CAST-PN PN for r;
     For each transition t in PN, do
        Let precond be the precondition corresponding to t;
        If precond is a fail or wait mode, then
           Add (precond, P, r, t) into HelpNeeds;
        Let effect be the effect of corresponding to t;
        Add (effect, P) into HelpEffects;
```

The online algorithm refers to the team organization and finds the current plan invocations. Through the offline algorithm, it infers all potential helping needs and coverage of the plan invocations. To identify the actual help needs, the online algorithm first filters the potential helping needs by the coverage implied by the current plan invocations. The online algorithm then has the agent communicate with agents needing help to decide the exact conditions for which help is needed. Through RoB-SMMs, agents can dynamically construct individual processes of agents needing help, but without concrete markings. That means, agents can be mutually aware of what actions needers are doing, except the exact time when the actions are being executed. Also, the potential help needs generated by the offline algorithm may contain variables and agents may not know their exact bindings. The potential helping agents know which roles need help and can decide which agents to communicate with according to their team organizations.

The online algorithm searches the plan library for a plan, and finds a team of agents to whom the roles in the plan can be delegated. If such a plan and team exist, the online algorithm coordinates with the team to invoke the plan. Each agent ag executes the following online algorithm:

```
PHB_Online()
  While ag is alive, do
    For each plan invocation P in ag's team organization, do
      For each helping need Need in HelpNeeds generated by
            PHB_Offline(P), do
        Bindings = Identify_Actual_Help_Needed(P, Need);
        If Bindings ≠ null, then
          Let HelpGoal be the predicate in Need with Bindings;
          Search for a plan P1 that achieves HelpGoal;
          If P1 ≠ null, then
            Search for a team G to invoke P1;
            If G≠ null, then
              Notify the help needers that Need has been helped;
              Ask G start P1;

Identify_Actual_Help_Needed (P, Need)
  Let (Pred, P, G, t) = Need;
  If Need in P has been helped, then return null;
  For each plan invocation PI in ag's team organization, do
    For each coverage Effect in HelpEffects generated by
        Helping_Offline(PI), do
      If the predicates in Effect implies pred, then return null;
  Check ag's team organization & find the agents T assigned to G;
  Ask one agent in T for the Bindings in Pred;
  Return Bindings;
```

For simplicity, we did not include the recognition of helping needs for backup behaviors in the algorithm. As explained earlier, this type of need can be identified by tracking plan termination conditions. We can enable backup behaviors by slightly changing the algorithms, but omit doing so here due to length considerations.

We used a pull mode of communication to identify actual help needs in the online algorithm. The communication volume and response time are decided by the frequency. The more frequently agents communicate with each other for identifying actually help needs, the more responsive agents are to help needs. Users can set the frequency of identifying needs according to the time requirements of specific domains.

5 Experiment and Analysis

We have constructed experiments on a multi-agent extension of the Wumpus World with a team T of three agents ag1, ag2 and ag3. Agent ag1, ag2 and ag3 play the roles of, a sniffer to sense wumpuses and gold in squares within a radius of two, a carrier to collect gold, and a fighter to shoot wumpuses, respectively. The goal of the team is to collect as much gold as quickly possible. Agents ag1 and ag2 form a subteam to invoke plan *Sense&Collect*, by which ag1 walks around and senses gold and wumpuses. and ag2 collects the gold found. Ag3 invokes plan *Wander*, by which ag3 just randomly moves in the map. Although ag3's plan *Wander* is irrelevant to the team goal, it could be replaced with any plan relevant.

Figure 1 shows the map we use. It contains 400 squares (20 by 20), and has 40 wumpuses and 60 pieces of gold. A piece of gold may be surrounded by wumpuses. Such gold is unreachable unless a path is opened by killing one of the wumpuses. Also, a piece of gold may be in a square together with a wumpus. Such gold is also unreachable unless the wumpus is killed. There are 25 pieces of unreachable gold in the wumpus world shown in Figure 1.

To illustrate the impact of proactive helping behaviors, we ran two configurations on RoB-CAST, with and without PHB, and collected team performance for each. The agents with and without PHB behave differently in two situations: 1) once ag2 knows a piece of gold surrounded by wumpuses, ag2 tries to find a path to reach the gold; or 2) once ag2 knows a piece of gold with a wumpus, ag2 tries to move to the square to collect the gold. Without PHB, nobody kills the wumpus and ,ag2 gives up collecting the gold. With PHB, ag3 can be aware of ag2's help need.. If ag3 knows it can help ag2 by killing a wumpus, then ag3 provides help by starting a plan *kill* to kill the wumpus.

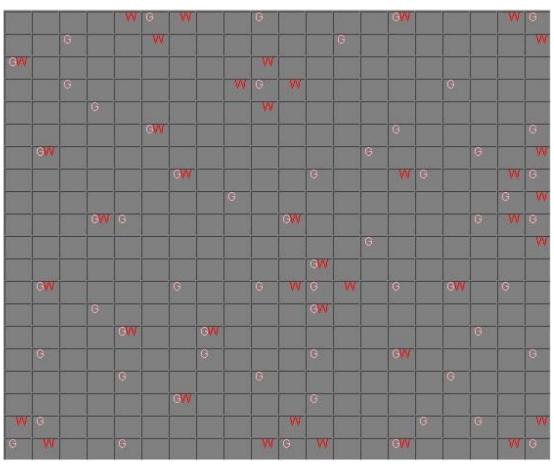

Figure 1. The wumpus world used in our experiment

The result of the experiment is shown by Figure 2. On average, the agents with PHB collected 54 pieces of gold while the agents without PHB only collected 35 pieces of gold. Moreover, to collect a certain amount of gold, the agents with PHB took less time than the agents without PHB. We note that the agents with PHB did not collect all gold and some pieces of gold were not collected even though proactive helping behaviors made them reachable.

As a trade-off for proactive helping behaviors extra communication is required to identify the needs. We set the frequency of checking help needs to be once per second. The agents responded help needs promptly and invoked plan kill 25 times (same as the amount of unreachable gold). The extra communication was about 1500 messages.

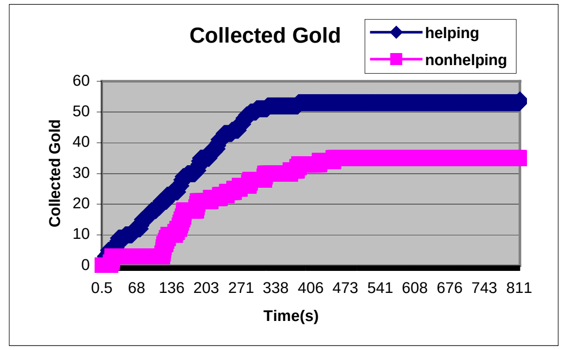

Figure 2. The distribution of the number of pieces of gold collected over the execution time

In summary, the experiment shown that proactive helping behaviors can improve team performance at the cost of a modest amount of communication.

6 Conclusion and Further Improvements

In summary, we have developed a formal model to enable proactive helping behaviors in teamwork. The model can identify two types of help needs (backup and promotion) and have agents initialize courses of actions to meet the needs. We have implemented the model in our teamwork architecture RoB-CAST. Experiments have shown that proactive helping behaviors improved team performance. Although the formal model is presented and implemented based on our role-based shared mental models, the formal model works on other shared mental models that enable mutual awareness of team processes.

Further improvements can be made. First, the model can be extended to monitor help needs during the execution of proactive helping behaviors. If a help need is dismissed, there is no need for agents to continue execution of helping behaviors. In our experiment, after ag2 waited for a period of time and gave up, the help need for collecting the gold was dismissed. However, ag3 still finished plan kill for the (no longer needed) help.

Second, the model does not capture the impact of helping behavior on what the agents are currently doing. Helping behaviors benefit the agents who need help, but they may hurt what agents are doing. For example, providers of help may have other higher priority of tasks; or the plan used for helping may lead to some effects that reverse the goals, which some agents are achieving. Also, multiple plans with different costs might usable to provide helping behaviors. An extension to weight the costs, side effects and benefits of these plans and apply theoretic decision-making to decide whether to provide helping behaviors or which plan to use would be useful.

Finally, complex planning mechanisms might be adopted to replace the mechanism of searching for a plan and team.

7 Acknowledgements

This research was supported by a DOD MURI grant F49620-00-1-0326 administered through AFSOR.

8 References

[1] Brehm, S., and Kassin, S. M. “Social Psychology” (Second Edition). Boston: Houghton Mifflin. 1993

[2] Cao, S. “Role-Based and Agent-Oriented Teamwork Modeling”. Texas A&M University, College Station, Texas, 2005.

[3] Cannon-Bowers, J. A., Salas, E., and Converse, S. A. “Shared Mental Models in Expert Team Decision Making”. Individual and Group Decision Making, Castellan, NJ, 1993, pp. 221-246.

[4] Cannon-Bowers, J. A., and Salas, E. “Reflections on Shared Cognition”. Journal of Organizational Behavior, 22, 2001, pp. 195-202.

[5] Cohen, P. R., and Levesque, H. J. “Teamwork”. Nous: Special Issue on Cognitive Science and Artificial Intelligence, 25(4), 1991, pp. 487-512.

[6] Grosz, B., and Kraus, S. “Collaborative Plans for Complex Group Actions”. Artificial Intelligence, 86(2), 1996, pp. 269-357.

[7] Jennings, N. R. “Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems”. The Knowledge Engineering Review, 8(3), 1993, pp. 223--250.

[8] Jennings, N. R., and Mamdani, E. H. “Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments”. 10th National Conference on Artificial Intelligence (AAAI-92), San Jose, CA, 1992.

[9] Jennings, N. R., Mamdani, E. H., Laresgoiti, I., Perez, J., and Corera, J. “GRATE: A General Framework for Cooperative Problem Solving”. IEEI-BCS Journal of Intelligent Systems Engineering, 1(2), 1992, pp. 102-114.

[10] Klimoski, R., and Mohammed, S. “Team mental model: Construct or metaphor?” Journal of Management, 20, 1994, pp. 403-437.

[11] Lind, G. “How moral is helping behavior?” American Education Research Association (AERA), Chicago, IL, 1997.

[12] Mathieu, J. E., Heffner, T. S., Goodwin, G. F., Salas, E., and Cannon-Bowers, J. A. “The Influence of Shared Mental Models on Team Process and Performance”. Journal of Applied Psychology, 85(2), 2000, 273-283.

[13] Miceli, M., Cesta, A., and Rizzo, P. “Autonomous help in distributed work environments”. Seventh European Conference on Cognitive Ergonomics, Bonn, Germany, 1994.

[14] Porter, C. O., Hollenbeck, J. R., Ilgen, D. R., Ellis, A. P. J., West, B. J., and Moon, H. K. “Backing up Behaviors in Teams: The Role of Personality and Team Structure”. Journal of Applied Psychology, 88, 2003, pp. 391-403.

[15] Rao, A. S., and Georgeff, M. P. “Modeling Rational Agents within a BDI-Architecture”. In Proceedings of the 2nd International Conference on Principles on Knowledge Representation and Reasoning (KR'91), Cambridge, MA, 1991.

[16] Rouse, W. B., Cannon-Bowers, J. A., and Salas, E. “The Role of Mental Models in Team Performance in Complex Systems”. IEEE Transactions on System, Man and Cybernetics, 22(6), 1992, pp. 1296-1308.

[17] Tambe, M. “Towards Flexible Teamwork”. Journal of Artificial Intelligence Research, 7(1), 1997, 83-124.

[18] Yen, J., Yin, J., Ioerger, T. R., Miller, M. S., Xu, D., and Volz, R. A. “CAST: Collaborative Agents for Simulating Teamwork”. 17th International Joint Conference on Artificial Intelligence (IJCAI'2001), Seattle, WA, 2001.