Agent-Based Team Aiding in a Time Critical Task

Terry R. Payne[†], Terri L. Lenox[‡], Susan Hahn[‡], Michael Lewis[‡] & Katia Sycara[†]

[†]*Carnegie Mellon University*
The Robotics Institute
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
{Terry.Payne|Katia.Sycara}@cs.cmu.edu

[‡]*University of Pittsburgh*
School of Information Sciences
135 N. Bellefield Ave.
Pittsburgh, PA 15260 USA
{tll|hahns|ml}@lis.pitt.edu

Abstract

In this paper we evaluate the effectiveness of agent-based aiding in support of a time-critical team-planning task for teams of both humans and heterogeneous software agents. The team task consists of human subjects playing the role of military commanders and cooperatively planning to move their respective units to a common rendezvous point, given time and resource constraints. The objective of the experiment was to compare the effectiveness of agent-based aiding for individual and team tasks as opposed to the baseline condition of manual route planning. There were two experimental conditions: the Aided condition, where a Route Planning Agent (RPA) finds a least cost plan between the start and rendezvous points for a given composition of force units; and the Baseline condition, where the commanders determine initial routes manually, and receive basic feedback about the route. We demonstrate that the Aided condition provides significantly better assistance for individual route planning and team-based re-planning.

1. Introduction

Emergency response tasks, both military or civilian, are characterized by environmental uncertainty, stress, and time criticality of decision making. The decision making process is distributed across different team members with different expertise, who are distributed in space and time, and who act with incomplete information in an uncertain environment. Hence, high quality computer assistance is critical. Recently, the technology of software agents has emerged as a suitable metaphor for interacting with computer processes that assist human decision making. Such software agents can reduce the amount of interaction between humans and the computer system and allow the humans to concentrate on other activities, such as assessing the situation, making decisions, or reacting to changes in the system [12]. In addition, such agents should not only retrieve information on request; but they should actively and intelligently anticipate, adapt and actively seek ways to support users [1,10].

When interacting with a computer, some metaphor should be adopted to guide the actions and expectations of the user [4]. The metaphor presented in this paper treats the computer as an intermediary that responds to user requests. Instead of simply entering commands or selecting objects from a GUI environment, the user and the computer both initiate communication, execute tasks and monitor their respective performances. This agent metaphor has been referred to as indirect management [4,5]. Interaction with traditional computer interfaces (e.g., programming or scripting languages) can be arbitrarily complex for humans and hence time-consuming and error prone. An agent metaphor can be a very powerful one especially when flexibility is desired by users who do not wish to (or cannot) explicitly instruct the computer [6].

However, the benefits of software agents may be undermined by an increase in complexity and resulting confusion when interacting with the agent. New skills, such as task decomposition and delegation, may be required to interact with sophisticated software agents or agent communities [11]. Conversely, those agents which shield us from complex interactions by quietly looking over our shoulders to anticipate our actions may actually decrease our situational awareness and leave us uncertain as to what is being done on our behalf [4]. It is important that users are able to construct their own goals and values, then decide, plan and act in ways to help these achieve goals and values. User autonomy can be reduced when users fail to understand what is happening within a system, when they cannot control the system, or when the agent and/or system behavior is unpredictable. Deskilling may also occur when the agent makes decisions for the user rather than just providing advice, or if the user is prevented from make the wrong decisions [3,8]. These difficulties can be compounded where multiple agents and humans are required to work as a team. Under such conditions, cascading delegation among software agents and passive assistance may complicate the already challenging task of cooperating, communicating, and monitoring the other human team members.

Our research focuses on agent-based decision aids. While much of the early focus on decision aids supported the individual [7], we focus on the middle ground of individually controlled software agents used in team tasks. Although it may be desirable to organize individuals into teams and provide support via software agents, this is not necessarily an easy task. The design of tools that aid human and computer members of a team should build upon the fundamental principles developed by computer scientists and psychologists over the past several decades, augmenting these fundamentals with an understanding of the special requirements of human-agent interaction. What roles should agents play in the overall team context? How can software agents help with the vast array of information available to the team without restricting their situational awareness? What are effective ways for software agents to interact with the human team members and with each other so as to increase team effectiveness? How can we indirectly manage agents? How should we structure communication so agents can communicate their expertise in an understandable way? How can we avoid (or at the very least detect) incorrect inferences [9]?

In this paper, we attempt to address some of these issues by comparing team decision making in a *Baseline* condition, where the humans do not have the advantage of intelligent decision aiding, and in the *Aided* condition. The paper is organized as follows: Section 2 addresses the different types of information that may be available to the human team members (i.e. commanders) and to the agents. The *Baseline* and *Aided* conditions are described in Section 3, and the experimental methodology outlined in Section 4. In Section 5, the results of the experiments are presented and discussed, while the conclusions and discussion of future work appear in the final section.

2. Using the Infosphere to make plans

When planning, humans face complex, dynamic environments in which they often lack sufficient knowledge, skills, and time to perform the tasks. There are issues of time pressure, conflicting subgoals, division of labor amongst subordinates, and allocation of resources within an evolving dynamic environment. For example, military commanders have a vast array of information at hand; this includes physical characteristics of the terrain, the location of any enemy forces that may be present, and the types, numbers, and capabilities of both their own and the enemy forces. In addition, they are also aware of specific objectives for their mission, as well as being highly trained and thoroughly briefed on doctrinal constraints. This information is part of the commander's infosphere. Information within the infosphere has the potential for data fusion, situation visualization, and "what-if" simulations. The volume of data may be large,

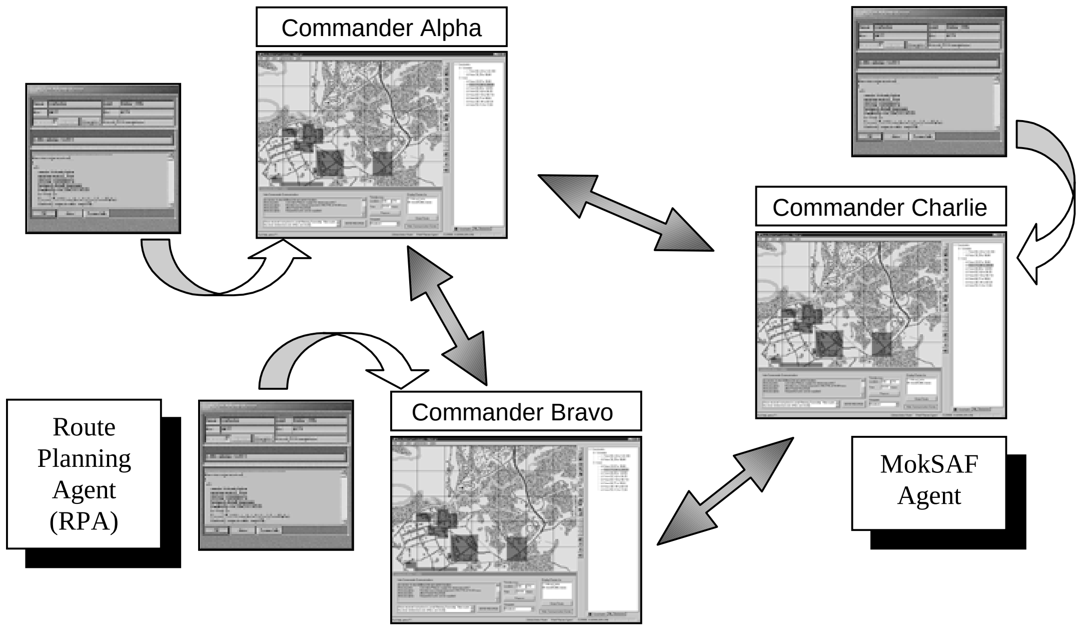


Figure 1. MokSAF Environment – Each MokSAF agent can communicate with other MokSAF agents, and with a single *Route Planning Agent (RPA)*

and is generally quantitative. Planning a successful mission, however, involves additional qualitative information that is difficult to make precise. Military commanders, like other decision-makers, have vast experiential information that is not easily quantifiable. This extra-infosphere data consists of intangible or multiple objectives involving morale, the political impact of actions (or inaction), intangible constraints, and the symbolic importance of different actions or objectives. Commanders must deal with idiosyncratic and situation-specific factors such as non-quantified information, complex or vaguely specified mission objectives and dynamically changing situations affected by incomplete/ changing/new information, obstacles, and enemy actions.

Software agents can plan, criticize, and predict the consequences of actions using the information from the infosphere with a greater accuracy and finer granularity than the human commanders. Multiple agents can be designed to cooperatively utilize the information in the infosphere to satisfy specified goals. However, the agents cannot anticipate or comprehend additional information, especially qualitative information. If agent-based aiding is to be effective, there should be ways for commanders participating in a planning task, to translate these intangible constraints into physical ones to interact with planning agents. The inclusion of intangible constraints raises a further question: how should software agents interact with their human team members to effectively incorporate these intangible constraints into their model of the physical environment?

The research reported in this paper addresses these issues. We have developed software agents that interact with human team members in a joint mission-planning task. We have also developed techniques for allowing the human commanders to express intangible constraints to the agents through the use of an appropriate graphical interface.

3. The *MokSAF* Environment

A computer-based simulation called MokSAF has been developed to allow two or more humans (acting as military commanders) to collaborate via two or more interface agents with one another when planning missions. A mission plan consists of one or more platoons of heterogeneous units, an agreed rendezvous time and location, and a set of routes for each platoon. The platoons start from different points, and each route ends at a common rendezvous. Each commander is responsible for the composition of one of the platoons, and for determining the route taken by that platoon. The commanders determine the individual routes and platoon compositions via MokSAF interface agents and *Route Planning Agents* or *RPA*s (See Figure 1). In addition, the MokSAF agent allows each commander to share routes with other MokSAF agents, and hence a commander can display the different routes on a single MokSAF user interface (See Figure 2).

MokSAF is loosely based on a virtual battlefield simulation called MODSAF (MODular Semi-Automated Forces). Although MODSAF is a rich simulation environment, the training and knowledge requirements are beyond what can easily be provided to the participants as part of this research. Figure 2 shows part of the MokSAF interface agent, including the terrain map and the toolbar. The terrain consists of soil (plain areas), roads (solid lines), freeways (thicker lines), buildings (black dots), rivers and forests. The rendezvous point is a red circle (upper left) and the start point is a yellow circle (lower right) on the terrain map. The primary route, either created manually by the commander in the *Baseline* condition or with assistance of the *RPA* in the *Aiding* condition, is shown in bright green, whilst other routes appear in muted colors.

The problem that the team of commanders has to solve is as follows: each commander must select appropriate vehicles to constitute his/her platoon so that:

(1) The platoon should reach the shared rendezvous point without running out of fuel.
(2) The route taken by each platoon should consume the minimum volume of fuel possible.
(3) The platoon should visit certain mid-points en-route.
(4) The route should not violate any physical constraints (such as crossing densely forested areas with large vehicles).
(5) The route should not violate intangible constraints (where an intangible constraint might specify *“avoid entering this specific area as it is a suspected minelfield”).*
(6) The combined platoons should contain a minimum subset of specified units at the rendezvous.

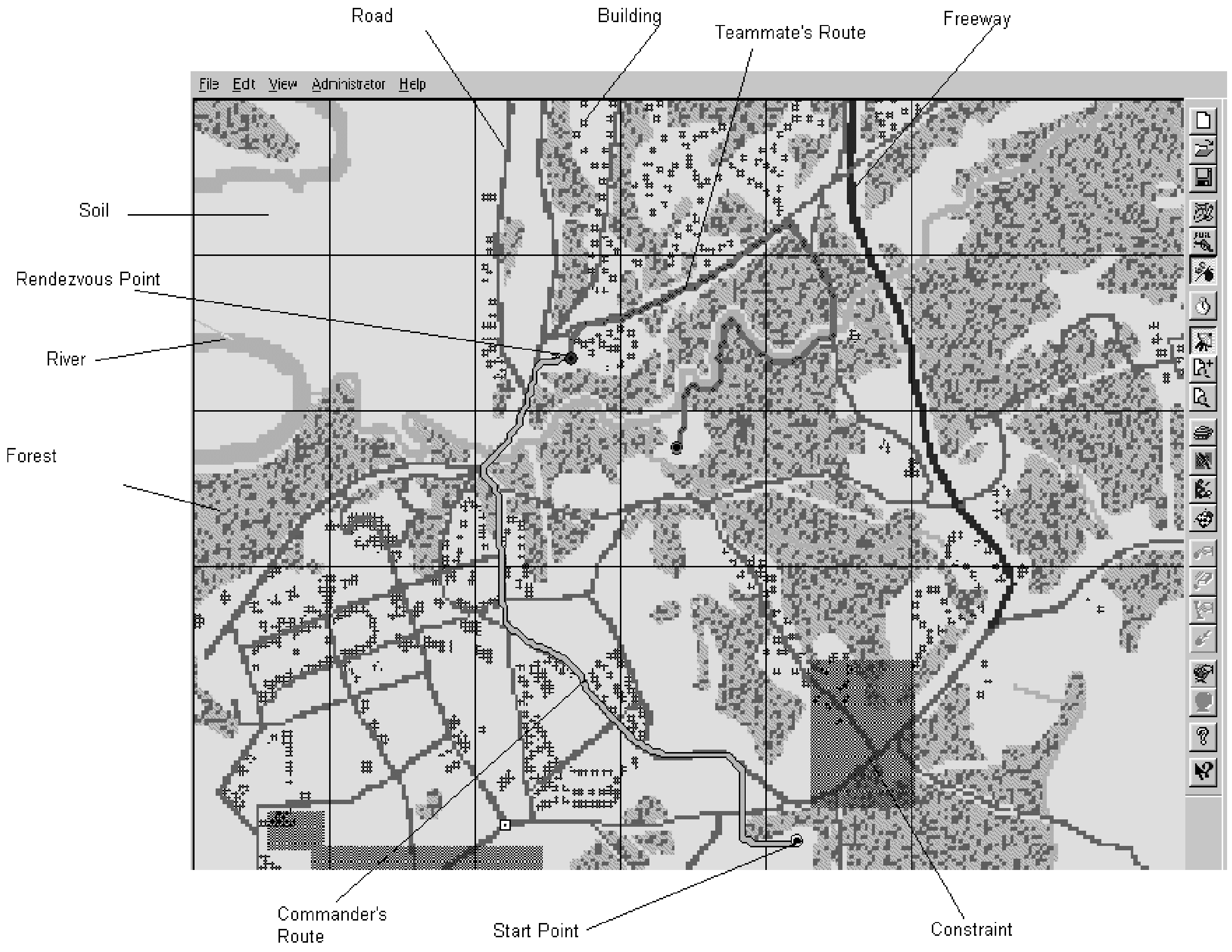
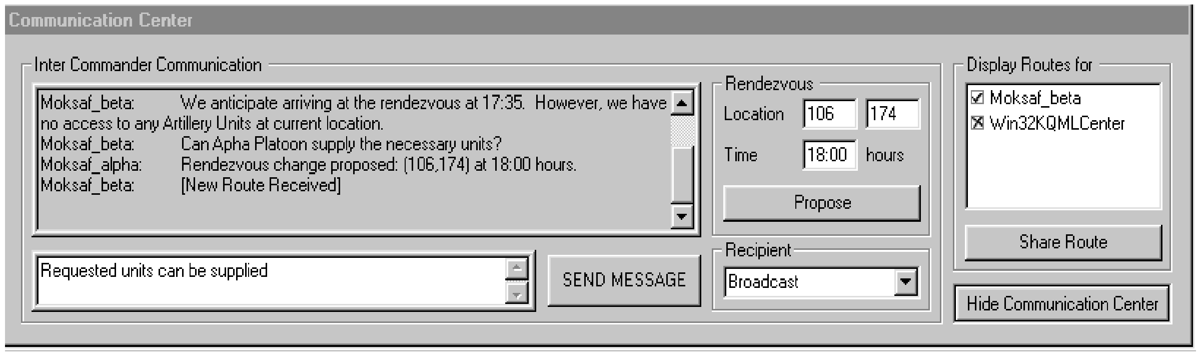


Figure 2. The MokSAF Agent Interface.

This is a complex constraint optimization problem. Each vehicle has different characteristics with respect to the types of terrain it can traverse and moreover, the vehicle's speed and fuel consumption depends on the type of terrain being crossed at a given time interval. In addition, the intangible constraints must be somehow represented so they can be taken into consideration during problem solving. Finally, the problem is of large scale since the planned route can be off-road i.e. vehicles traverse open spaces, such as desert, grassy areas, or forests (without the advantage of having marked roads to constrain the search).

This task has a variety of characteristics, some of which are easy for humans to deal with and some that are difficult. What makes the task easy for humans is its visual nature, namely the fact that routes can be drawn on the map. In contrast, it is very difficult for humans to calculate path lengths, vehicle speeds or fuel consumption. These latter characteristics make the task more amenable to computerized aiding.

Task analysis, i.e. the analysis of various components of a task to determine whether or not human interaction is better suited to solving the task than agent interaction inspired our solution to the problem of encoding and presenting *intangible constraints* to the *RPA*. If computerized aiding is to be effective, intangible constraints, which may be transient and unstructured (such as "*when on an exercise avoid routes that go near schools during term time, unless the platoon consists of light vehicles*") should be encoded in a form that can then be utilized by the agents when assisting with the planning. Some form of feedback should also be provided so that the commander can verify that the encoding is correct. To resolve this problem, *intangible constraints* are represented by shaded rectangles drawn on the map (see Figure 2). These regions represent areas that the platoon should avoid. This is a visual representation that is easy



Moksaf_beta: We anticipate arriving at the rendezvous at 17:35. However, we have no access to any Artillery Units at current location.
Moksaf_beta: Can Apha Platoon supply the necessary units?
Moksaf_alpha: Rendezvous change proposed: (106,174) at 18:00 hours.
Moksaf_beta: [New Route Received]

Requested units can be supplied

Location: 106 174
Time: 18:00 hours
Recipient: Broadcast
Display Routes for: [✓ Moksaf_beta] [✓ Win32KQMLCenter]

Figure 3. The MokSAF Communication Center.

for the user to both perform and verify. Once drawn, these constraints can be shared with the *RPA*, which can utilize this knowledge when providing assistance with future routes.

The *Route Planning Agent* (*RPA*) utilizes an off-road route-planning algorithm based on Dijkstra's shortest path algorithm [2] to provide assistance in the *Aided* condition. This is used to determine the minimum cost route between two points (a start and rendezvous location), given a terrain map of a geographic region and the characteristic behavior of a given platoon for each of the different terrain types. A traversal cost, which is assigned to each pixel on the map, is generated by mapping the terrain type for that pixel with the speed characteristics of a given

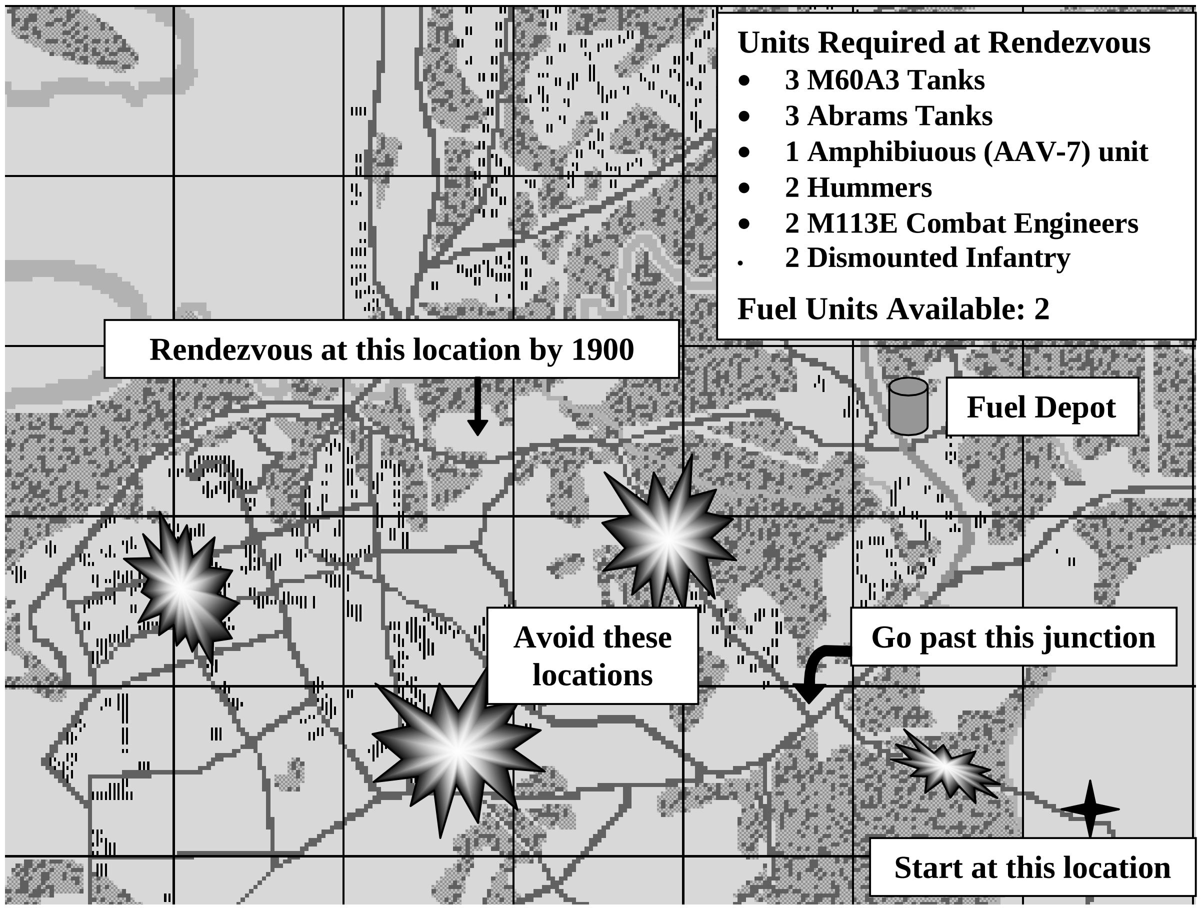


Figure 4. Mission-Brief Map. Includes regions that should be either visited or avoided.

platoon. The speed characteristics of a platoon are determined by finding the lowest speed characteristics of the units within that platoon (i.e. the platoon may travel no faster than its slowest unit). As certain units may only traverse certain terrain types, this may also constrain the route of the final path. The *RPA* is also aware of any intangible constraints that have been graphically encoded by the commanders on the MokSAF interface agent. The commanders also express simultaneous goals in the form of multi-attribute utility functions for the agents and may need to adjust these constraints to devise solutions to satisfy these utilities. For example, if the commander wishes to avoid a certain area or knows that a fuel depot is being used by another commander, then these additional constraints should be encoded (as intangible constraints on the MokSAF agent) and the *RPA* instructed to replan a new route.

In the *Baseline* condition, the commander is responsible for determining the route and drawing it on the MokSAF agent, by specifying an arbitrary number of individual points on the map along the desired route. Straight-line segments are determined to connect these points and generate the route. The route is then sent to the *RPA*, which checks to see whether the route violates any physical constraints or encoded intangible constraints, and estimates fuel consumption for the commander’s platoon. If such constraints have been violated, or if the platoon has insufficient fuel to complete the journey, the commander is notified, and can then modify the route or the composition of the platoon. This process continues in an iterative fashion until the commander is satisfied with the route.

The MokSAF communication center (Figure 3) is typically displayed below the terrain map. It provides the facility for commanders to communicate with each other, and maintains a history of these communications for later reference. Each sent message is annotated with the name of the issuing commander. Commanders can elect whether to “broadcast” their messages to all the other commanders on the team or whether to send the message to a specific teammate. Messages consist of textual dialog pertaining to the commander's plans, negotiations regarding the allocation of units, recommendations for suggested changes to the rendezvous location and/or time, and other requests for information. In addition, commanders may also share their latest routes with other commanders. These shared routes can then be superimposed on the individual maps (as illustrated in Figure 2).

4. Methodology

In the current MokSAF experiments, the planning task is deliberative, iterative and flexible. There are three commanders, each of which has a different starting point but share a common rendezvous point. The commanders must coordinate the number and types of vehicles they plan to move from the individual start points to the rendezvous point. The mission briefing supplied to the commanders (Figure 4) provides them with a list of vehicles that should arrive at the rendezvous point. In addition, the commanders are instructed to avoid generating routes that lie on the same path as any other commander, and that they should coordinate their routes through the communication center to avoid this. Each commander selects units for his/her platoon from a list of available units. Units currently available are M60A3 tanks, M109A2 artillery units, M1 Abrams tanks, AAV-7 amphibious assault vehicles, HMMWVs (i.e., hummers), ambulances, combat engineer units, fuel trucks and dismounted infantry. Commanders have 15 minutes to determine the composition of their platoon, and plan a route from a starting point to the rendezvous point for that platoon. Once a commander is satisfied with the individual plan, he/she can share it with the other commanders and resolve any conflicts. Conflicts can arise due to shared routes, shared resources, or the inability of a commander to reach the rendezvous point at the specified time.

The experiments were performed to investigate a number of hypotheses. Can agent-based assistance assist in the completion of team tasks? If assistance is provided in achieving the individual goal, then does this improve the quality of the team goal? Are intangible constraints suitable for encoding and sharing intangible constraints with different agents? Does agent-based aiding become more effective as the complexity of the intangible aspects of a planning problem increase?

4.1. Materials

MokSAF 2.0 was used for this pilot study. It consists of the standard terrain map and markings, a toolbar as illustrated in Figure 2, a communication center where commanders can send and receive messages and share plans (Figure 3), and a constraint tree. Two experimental conditions were used; the *Baseline* condition in which the routes are determined manually, and the *Aided* condition, in which the *RPA* determines the routes.

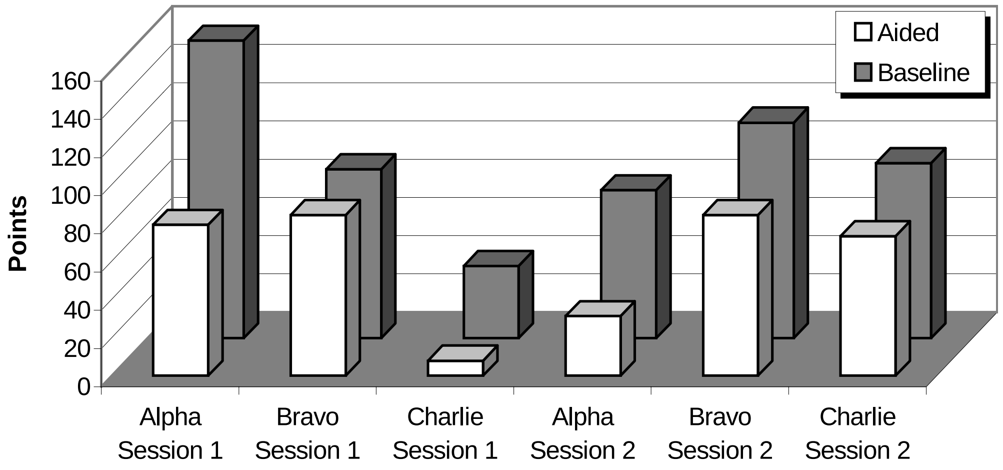


Figure 5. Mean Route Lengths at Session End.

4.2. Participants

Fifteen three-person teams were recruited (10 teams in the *Aided* condition and five in the *Baseline* condition) from the University of Pittsburgh and Carnegie Mellon University communities. Participants were recruited as intact teams, consisting of friends or acquaintances. Teammates needed to communicate with one another to complete their tasks successfully.

4.3. Procedures

Each team participated in a 90-minute session that began with a 30-minute training session in which the MokSAF environment and team mission were explained. Each team member was assigned the role of one of three commanders (*Alpha*, *Bravo* or *Charlie*). The composition of each platoon was heavily dependent on the platoon's start position. For example, a commander may only successfully leave an island if the commander's platoon only consisted of either amphibious vehicles or dismounted infantry. The team was told to find the optimal path between the start and rendezvous points, to avoid certain areas, to go by specific areas, to meet the mission objectives for numbers and types of units in their platoon, and to avoid joint use of paths with the other commanders. After the training session, the team participated in two 15-minute trials. Each trial used the same terrain, but different start and rendezvous points and different platoon requirements. At the conclusion, participants were asked to complete a brief questionnaire.

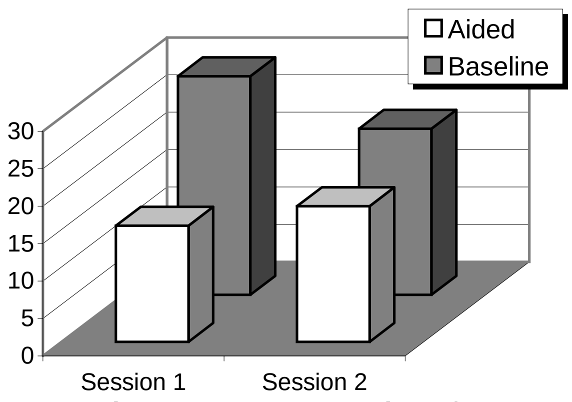


Figure 6. Average Number of Feedback Activations.

5. Results

The mean lengths of the routes generated within the *Baseline* condition by each commander were generally longer than those generated within the *Aided* condition for both sessions[1]. The graph below (Figure 5) illustrates how the routes that were shared at the end of the experiments differed between each of the commanders. The difference in route lengths varied significantly, from a difference of approximately 5 points for Bravo (Session 1) to 77 points for Alpha (Session 1). Figure 6 compares the average number of times the *RPA* was used (i.e. activations) for the two conditions. In the *Baseline* condition, the *RPA* was activated more times than within the *Aided* condition. It also took longer for routes created in the *Baseline* condition to be shared amongst team members (5min 26sec & 5min 38sec for Sessions 1 & 2) compared to the *Aided* condition (2min 56sec & 2min 19sec minutes respectively) as illustrated in Figure 7. These results suggest that commanders were able to identify faster (and more economic) routes with the *Aided* condition, requiring

[1] The route length reflects the total number of points that describe a route, and should not be confused with the number of mid-points provided by a commander when constructing routes within the *Baseline* condition.

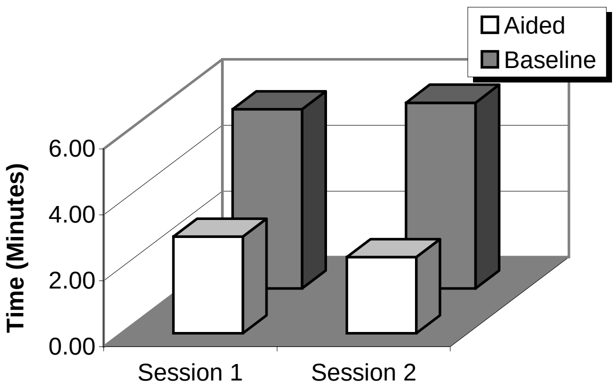


Figure 7. Time At Which Route Was First Shared.

fewer interactions with the *RPA*, and hence could be shared faster with team mates. These results support the hypothesis that individual plans can be generated more efficiently if greater assistance is provided to the commanders.

The individual path lengths for each commander were measured when routes were first shared with the team and at the end of the 15-minute trial. The lengths of the initial shared routes were expected to vary with respect to those at the end of each trial, when the routes taken by other commanders and the overall team goal was considered. However, there was little change overall in the mean length of the routes generated within the *Aided* condition (Figure 8). This contrasts sharply with the change in route lengths for the *Baseline* condition; there was a significant difference in the change in route lengths from when the routes were first shared and at the session end ($p < .018$). This difference may be due to the quality of the route was in when first shared; i.e. the routes drawn in the *Baseline* condition may have required further refinements during the trial, than those generated by the *RPA*. Both approaches required refinements to the original routes; this was due in part to the interactions with teammates. However, there was no difference in the selection of units in either experimental condition. Also, none of the experimental groups succeeded in reaching the rendezvous point with all the required units, and hence successfully achieving the team goal. This suggests that commanders were poor at coordinating the selection of units, even when more time was available because of the assistance in route planning (i.e. the *Aided* condition).

6. Conclusions

The Route Planning Agent has been shown to provide better decision support both for individual route planning and team-based replanning as part of the *Aided* condition when compared to the *Baseline* condition. The main difference between the two conditions was not the quality of the route in each Session, but rather the substantially more time that routes took to be constructed in the *Baseline* condition. The finalized coordinated routes were uniformly better for each of the individuals in the *Aided* condition group and also for the team as a whole. Despite this clear superiority, participants in this group frequently expressed frustration with the indirection required to arrange constraints in the ways needed to steer the planner's behavior and often remarked that they wished they could “just draw the route by hand”.

In the *Baseline* condition, subject complaints focused more closely on the minutiae of interaction. In its current form, the user “draws” a route using the MokSAF agent by specifying a sequence of points at a fixed resolution. This is achieved by specifying an initial or intermediate point in the path and then specifying a second point. A path segment is then drawn in a straight line between these

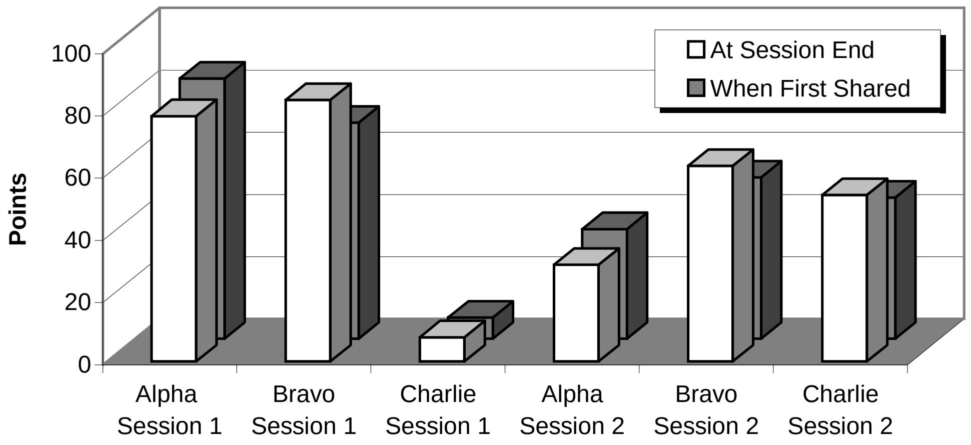


Figure 8. Mean Route Lengths in the Aided Condition.

points. A route is built up incrementally by piecing together a long sequence of such segments. Although there are other tools for editing this path (such as inserting new points within the route, repositioning or deleting existing points), the process of manually constructing a long route is both tedious and error prone. Routes generated by the *RPA* automatically avoid obstacles such as trees, and follow variations in certain terrain types such as roads. However, when a user constructs a manual route, low fidelity routes frequently violate terrain constraints (such as passing through buildings), and fail to follow optimal paths such as curves in roads. Although the inclusion of additional points into the route can overcome this problem, this process can be time consuming. Reducing the time of human planners to complete the task is obviously important in time critical tasks, such as mission planning. However, subjects spent more time refining their individual routes, as opposed to coordinating with other team members to improve the overall team task. Although our work was done in the domain of joint mission planning, we believe the results to be valid for similar tasks, such as in emergency response.

7. Acknowledgements

This research was supported by an Office of Naval Research grant N-00014-96-1-1222. Special thanks go to Constantine Domashnev, Glenn Lautenbacher and Martin van Velsen for the assistance in the development and testing of this software.

8. References

[1] Bradshaw, J. M. (1997). “Introduction,” in J. M. Bradshaw (ed.) *Software Agents*. AAAI Press: Menlo Park, CA, 3-48.

[2] Dijkstra, E.W. (1959). “A note on two problems in connexion with graphs.” *Numerische Mathematik*, 1:269-271.

[3] Friedman, B., and Nissenbaum, H. (1997). “Software agents and user autonomy,” Proceedings *of the First International Conference on Autonomous Agents (Agents ’97)*, ACM Press: New York, 466-469.

[4] Lewis, M. (1998). “Designing for human-agent interaction,” *AI Magazine*, Summer 1998, 67-78.

[5] Maes, P. (1994). “Agents that reduce work and information overload,” *Communications of the ACM,* 37(7), 31-40, 146.

[6] Payne, T. R and Edwards, P. (1997). “Interface Agents that Learn: An investigation of Learning Issues in a Mail Agent Interface” *Applied Artificial Intelligence*, 1997, Vol. 11(1), 1-32.

[7] Roth, E. M., Malin, J. T. and Schreckenghost, D. L. (1997). “Paradigms for Intelligent Interface Design.” In M. Helander, T. K. Landauer, and P. Prabhu (Eds) *Handbook of Human-Computer Interaction, Second Edition*, 1177 -1201.

[8] Sarter, N. B., and Woods, D.D. (1995). “From tool to agent: the evolution of (cockpit) automation and its impact on human-machine coordination,” *Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting*, 79-83.

[9] Sycara, K. P., and Lewis, C. M. (1991). “Cooperating of heterogeneous agents through the formation of shared mental models,” *Proceedings of the AAAI-91 Workshop on Cooperating Intelligent Systems*, July 1991.

[10] Sycara K., Decker, K., Pannu A., Williamson M., and Zeng D. (1996). “Distributed Intelligent Agents”, *IEEE Expert: Intelligent Systems and their Applications*, Vol. 11, No. 6, December 1996.

[11] Sycara, K., Decker, K. and Zeng, D. (1998). “Intelligent Agents in Portfolio Management". In N. Jennings and M. Woolridge (eds), *Agent Technology: Foundations, Applications, and Markets*. Chapter 14, Springer 1998, 267-283.

[12] Zachary, W., Le Mentec, J-C., and Ryder, J. (1996). “Interface agents in complex systems”, in *Human Interaction with Complex Systems: Conceptual Principles and Design Practice*, C. A. Ntuen and E. H. Parks (eds.). Kluwer Academic Publishers