The First International Workshop on

Mixed-Initiative Multiagent Systems (MIMS)

May 11, 2009

Budapest, Hungary

held in conjunction with AAMAS-2009 (the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems)

Workshop Proceedings (W12)

The First International Workshop on

Mixed-Initiative Multiagent Systems (MIMS)

May 11, 2009

Budapest, Hungary

Organization & Location

The first MIMS workshop is held in conjunction with AAMAS-2009 (the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems), in Budapest, Hungary. It is taking place on May 11, 2009, preceding AAMAS 2009.

Aims and Focus

Mixed initiative represents collaboration between humans and agents to benefit from the strength of both parties. While the initiation and maintenance of bilateral interaction between humans and agents in mixed environments is highly favorable, the process is certainly not trivial.

Many of the challenges associated with mixed initiative have been studied in recent years within specific targeted AI research communities such as interruption management and adjustable autonomy. The focused studies made great progress; however a unified system-wide approach, such in the case of intelligent user-interfaces is missing. For example, both interruption management and adjustable autonomy domains attempt to improve the process of initiating interaction with the user when both the agent and the user

have a joint goal. Nevertheless, while for interruption management the analysis focuses mainly on reasoning about the user's disturbance by the interruption, adjustable autonomy research mostly focuses in finding the best time from the system state of the problem to initiate interaction with the user. A combined approach in this case would attempt to find the best interaction timing, taking into consideration both aspects of the problem.

The MIMS workshop focuses on multiagent systems that interact with humans. Such an interaction can be done explicitly, i.e., direct human-agent interaction, or implicitly, i.e., interaction through emergent behavior techniques. In both cases, agents and humans, as individuals or as groups, can take initiative and decide what to do next. In such environments, humans and agents may share goals or have conflicting goals, and they may collaborate or compete for resources. The primary goal of this workshop is to bring together multiagent researchers from diverse backgrounds that looked at key issues in multiagent mixed-initiative systems in order to search for a synergy of ideas. In particular, discussion is encouraged in the following topic areas:

- Innovative approaches for initiating and managing interactions between agents and humans in collaborative environments.
- Human vs. agent in a multiagent environment: strengths and weaknesses.
- Evaluating the system-wide benefits of joint human and computer collaborations.
- Learning user preferences for making decisions on her behalf in mixed environments.
- Human and agent reactions to interruption and repeated interruptions.
- Combining interruption management and adjustable autonomy to a unified framework.
- State representation and visualization for enhanced agent-human interaction.
- Applications and case studies.
- Emergence, evolution and culture of mixed groups, teams and communities.
- Control protocols and philosophy: who's in charge?
- Building trust between agent and human. Norms and commitments in a mixed initiative environment.
- Verification and validation techniques and tools.
- Mixed-initiative architecture in human-robot environments.

Workshop Organization

Programme Chairs

David Sarne
Osher Yadgar

Programme Committee

Pauline Berry
Ece Kamar
Roger Mailler
David McSherry
Charlie Ortiz
Kanna Rajan
Zack Rubinstein
Neil Yorke-Smith

Table of Contents

Algorithm Steering for Mixed-Initiative Robot Teams

Dhruba Baishya and Michael Lewis
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15208
dhrubajbaishya@gmail.com, ml@sis.pitt.edu

Abstract. While much research has been devoted to human-robot interaction (HRI) with individually controlled robots or round-robin control for independently operating ones, the problems of controlling autonomously coordinating robot teams remain largely unexplored. Although MAS researchers have devoted significant effort to understand human interaction with teamwork algorithms other significant classes of coordination algorithms have not received comparable attention. In particular, human interaction with biologically inspired and optimizing control algorithms has been long neglected. These algorithms which are ideal for tightly coordinated tasks such as formation flying or simultaneous rendezvous require highly coordinated mutual adjustments yet have goals that can be simply specified. The problem arises when the operator wants the system to do anything else. Because there is little or no connection between the parameters available to the operator and the behavior that results we call such algorithms *opaque*. We conjecture that in many cases this problem may be relatively easy to solve and propose a taxonomy-in-progress to help identify classes of algorithms to be considered.

Keywords: multirobot systems, human-robot interaction, teamwork algorithms.

Applications for multirobot systems (MrS) such as interplanetary construction or cooperating uninhabited aerial vehicles will require close coordination and control between human operator(s) and teams of robots in uncertain environments. Human supervision will be needed because humans must supply the perhaps changing, goals that direct MrS activity. Robot autonomy will be needed because the aggregate demands of decision making and control of a MrS are likely to exceed the cognitive capabilities of a human operator. Controlling robots that must act cooperatively, in particular, will likely be difficult because it is these activities [7] that theoretically impose the greatest decision making load. Because some functions of a MrS such as identifying victims among rubble depend on human input, evaluating the operator's span of control as the number of controlled entities scale is critical for designing feasible human-automation control systems.

Current estimates of human span of control limitations are severe. Miller [11], for example, showed that under expected target densities, a controller who is required to authorize weapon release for a target identified by a UCAV could control no more than 13 UAVs even in the absence of other tasks. A similar breakpoint of 12 was found by [4] for retargeting Tomahawk missiles. Smaller numbers (3-9) [3] have typically been found for ground robots which usually require more frequent attention.

To extend operator span of control to larger teams we must consider how control difficulty for different control tasks grows with increases in team size. Computational complexity theory [13] offers one possible approach. Borrowing concepts and notation from computational complexity, authorization for weapon release after operator verification of each UAV-detected target, can be considered O(n) because demand increases linearly with the number of UAVs to be serviced. Anther form of control such as designation of an attack region by drawing a box on a GUI (Graphical User Interface), being independent of the number of UAVs, would be O(1). Practical applications are likely to require some mixture of control regimes. In our prior work with wide area search munitions [9], for example, the operator specified search and jettison areas and ingress and egress routes , O(1), but was also required to authorize attacks and allowed to command UAVs directly, both tasks of O(n) complexity. Examined from this perspective the most complex tasks faced in controlling large teams are likely to be those that involve choosing and coordinating subgroups of UVs. Simply choosing a subteam to perform a particular task (the iterated role assignment problem), for example, has been shown to be O(mn) [7].

The flip side of reducing command complexity for the human is to increase the reliance on automation to accomplish the task. The human's O(1) search command, for example, requires path planning, coordination, resource management, etc. from the robot team. The complexity of tightly coordinated tasks leads to a much shorter span of control than the O(n) tasks cited earlier. For a tightly coupled task such as box pushing (2 UGVs moving a box), for example, the operator is completely occupied commanding 2 robots [19] because every movement by one robot displaces the box requiring a corresponding movement by the other to catch up. For this reason for all but the simplest tasks coordination will need to be automated for robots to remain amenable to human control.

Relative to human control, multirobot coordination approaches can be divided into two general classes: teamwork algorithms that involve explicit assignment of roles and execution of plans and coordination schemes whose mechanisms are less cognitively accessible. In teamwork approaches such as Playbook [12], MissionLab [1], or Machinetta [16] the coordinated activities are planned out in advance often with the aid of graphical tools. Later during the mission these plans are executed by the robot team with human input limited to such things as calling plays (canned plans) or filling a role within a plan such as approving targets. While the moment to moment behaviors of the robots may involve complex interactions needed to coordinate movement the existence of a cognitively accessible plan governing their overall pattern of behavior provides the needed context for an operator to monitor and intervene as needed.

Many tightly coordinated tasks such as formation flying or simultaneous rendezvous require highly coordinated mutual adjustments yet have goals that can be simply specified. These tasks are probably best performed by algorithms based on optimization or biologically inspired control laws that can handle the extensive computation required. [10, 2], for example, have developed models for a human to assign tasks to a UAV cluster where the tasks are heavily constrained by the physical platform, such as the large turning radius of LOCAAS munitions, the sequence of activities that must occur for LOCAAS munitions, the timing constraints imposed by coordinated strike missions, and the physical relationships that must be enforced for certain types of search. The UAVs autonomously generate a plan that represents their constraints, and then present this plan to an operator who may influence the plan by changing costs, priorities, and constraints.

Because optimizing or biologically inspired control algorithms are very different from the cognitive processes a human might employ for the same problem, they can be difficult for human operators to comprehend or control. [15], for example, reported difficulties operators experienced in controlling a system making optimal weapons to target assignments because they could not designate targets directly but instead needed to adjust weights through trial and error to find a plan containing a desired target. This is a general problem of a class of algorithms we call opaque because their inner workings are not cognitively accessible to an operator. The difficulty of interacting through such algorithms is that while the primary goal for which they were designed, such as the region to be searched or rendezvous point, is easily expressed most other aspects of their behavior cannot be controlled directly. In both our examples control of other aspects of behavior required adjusting algorithm parameters in a way that could not be directly linked to the consequent behaviors.

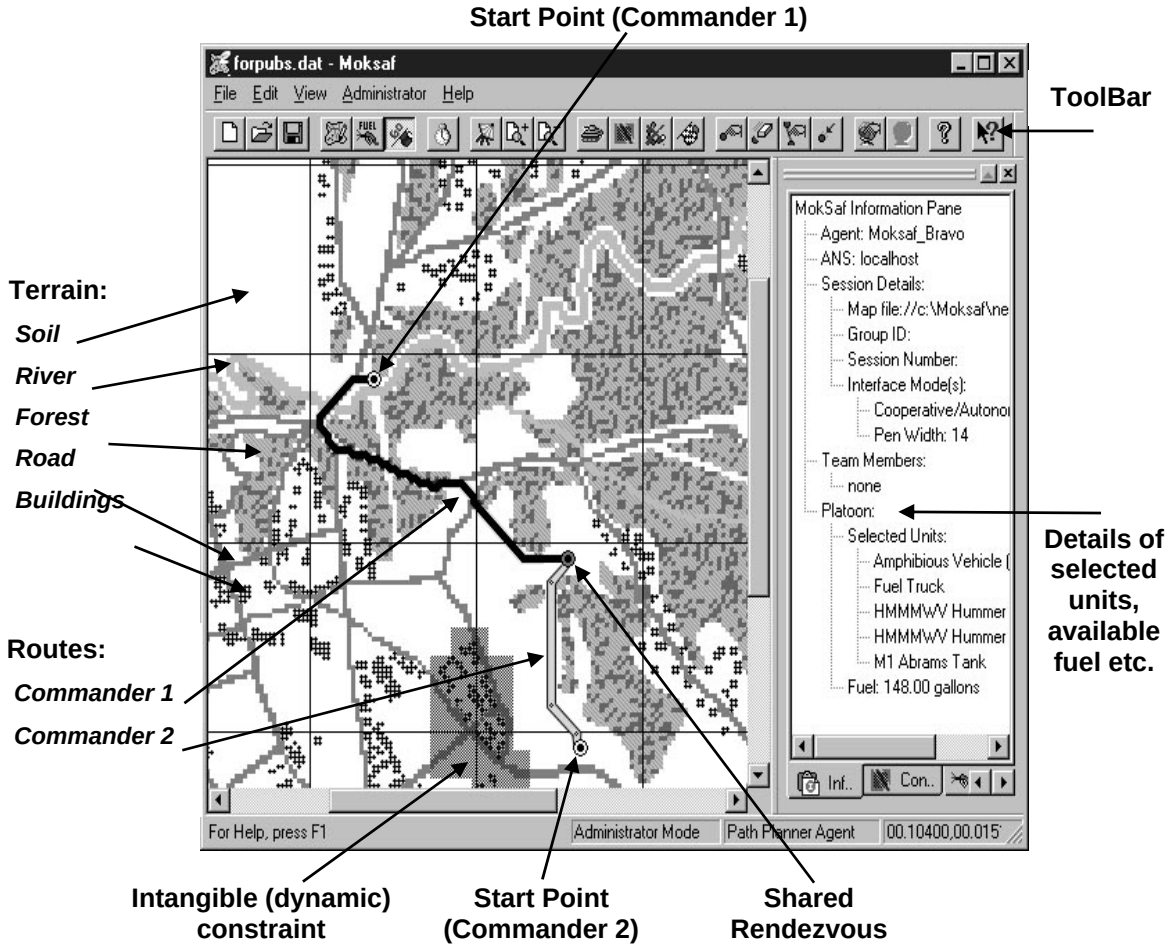
Robotics researchers studying multirobot control have frequently attempted to provide the operator with additional avenues of control usually through adjustments to parameters where a direct cognitive link could be found or imagined. Ron Arkin [1], for example, allowed operators to change the value of wanderlust, the magnitude of random deviations from a planned path the robot was allowed. Lynn Parker allowed the operator to adjust the twin tropisms impatience and acquiescence [14] to affect a robot's willingness to persist or abandon a role. One could imagine similar schemes involving biologically inspired local control laws such as broadcasting a change to a separation parameter in order to alter the dispersion of UAVs flying in formation. In each of these cases, however, the change in behavioral parameters does not directly impact the task but rather the way the robots perform the task. This opens the door for unanticipated consequences such as a wandering robot prevented by obstacles from returning to its path or a UAV formation that breaks apart due to sensing errors that grow at greater separations. What is needed for more effective human control is something akin to inverse kinematics that allows the operator to communicate the desired effect directly.

Because there are relatively few types of multirobot tasks that are both operationally relevant and admit optimizing or biologically inspired solutions (target assignment, parallel search, formation following, rendezvous, etc.) we believe these problems might be solved in a divide and conquer fashion. Our conjecture is that for any one of these task types there will be a finite and hopefully small number of *task relevant commands* an operator might desire to employ. In the weapon to target task, for example, task

relevant commands would include those referencing particular targets as well as global target types. To accommodate such commands the algorithm would need provisions to allow the operator to raise/lower the weight associated with a particular target rather than only for a target class. This adjustment would not require any significant change to the algorithm and could be presented transparently to the operator in terms of the domain as a command to either force inclusion or exclusion of targets. This simple change could have eliminated the difficulties experienced by Roth's subjects. We hypothesize that simple solutions of this sort may be frequent rather than rare and that commonly reported difficulties controlling opaque algorithms arise only because they have been written with a single goal without considering other actions an operator might desire.

MokSAF

The MokSAF path planning system based on Dykstra's algorithm, illustrates how an opaque algorithm can be effectively steered with minimal changes by paying attention to the task and task relevant commands. Human decision-makers, such as military commanders, face time pressures and an environment where changes may occur in the task, division of labor, and allocation of resources. Information such as



terrain characteristics, location and capabilities of enemy forces, direct objectives and doctrinal constraints

Figure 1. MokSAF Interface

are all part of the commander's "infosphere." Special purpose algorithms with access to this information can plan, criticize, and predict the consequences of actions at a greater accuracy and finer granularity than

the human commanders can. There is also, however, information that is not directly accessible to software. Such information includes intangible or multiple objectives involving morale, the political impact of actions (or inaction), intangible constraints, and the symbolic importance of different actions or objectives. When participating in a planning task, commanders must translate these intangible constraints into tangible ones to interact with planning algorithms.

We developed a computer-based simulation called MokSAF to evaluate how humans can interact with planning algorithms within a team environment. MokSAF is a simplified version of a virtual battlefield simulation called ModSAF (modular semi-automated forces). MokSAF allows two or more commanders to interact with one another to plan routes in a particular terrain. Each commander is tasked with planning a route from a starting point to a rendezvous point by a certain time. The individual commanders must then evaluate their plans from a team perspective and iteratively modify these plans until an acceptable team solution is developed.

The interface used within the enhanced MokSAF Environment is illustrated in Figure 1. The interface presents a terrain map, a toolbar, and details of the team plan. The terrains displayed on the map include soil (plain areas), roads (solid lines), freeways (thicker lines), buildings (black dots), rivers and forests. The rendezvous point is represented as a red circle and the start point as a yellow circle on the terrain map. As participants create routes with the help of a planner using Dykstra's algorithm, the routes are shown in bright green. The second route shown is from another MokSAF commander who has agreed to share a route. The algorithm is steered by the partially transparent rectangles representing intangible constraints that the user has drawn on the terrain map. These indicate which areas should be avoided when determining a route. As reported in [18, 8] this mixed initiative approach led to superior performance.

For this problem Dykstra's algorithm works efficiently by treating soil types as weights (asphalt=1 and water=infinity) so paths follow roads if they can, avoid open water and forests, etc. In terms of the single goal of shortest, lowest cost path it succeeds, however, there is no way to incorporate the intangible constraints known only to the commander. Upon considering the commander's task it is apparent that *task relevant* commands need to include excluding some areas from paths. This is easily achieved within the algorithm by artificially changing the soil weight of designated areas and easily used by the operator who can express his intent directly in the problem domain by marking exclusion areas on the map. In a close parallel to the weapons to targets example a simple but task informed modification to the algorithm overcame the problems of indirect control.

While three examples do not prove a principle we hope that by making our approach to re-engineering opaque algorithms explicit we can extend it to a larger class and come to understand for which types of algorithms it is appropriate.

As presently conceived the process consists of four steps:

1) identify objects in the domain subject to control; weapons, UAVs, and paths in our examples

2) examine the user's task and identify potential goals other than the focal goal of the algorithm

3) identify direct manipulation or other forms of command that might let the operator express these goals in a direct way in terms of domain objects

4) identify ways these commands might be realized within the algorithm. If none are available examine potential for transferring control over some assets to user.

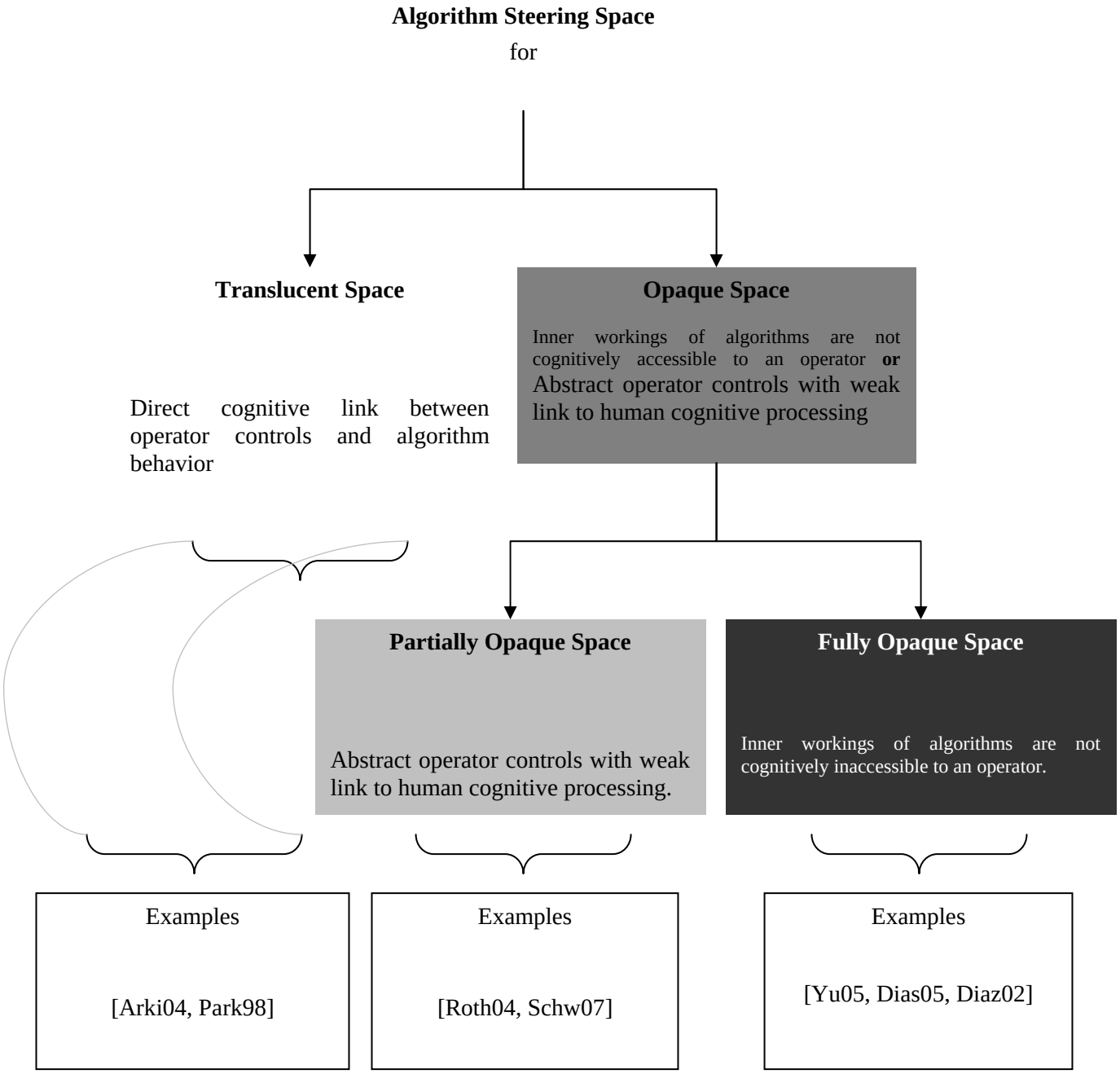
To begin our effort we are proposing a "strawman" taxonomy to help find representative cases to test.

A Possible Taxonomy for Algorithm Steering

We are proposing a user-based taxonomy for algorithm steering. This MrS Space is broadly classified into three categories –First, transparent space, where the operator can express goals explicitly in terms of domain objects. Second, translucent space, where some cognitive links exist between operator controls and algorithm behavior. Third, opaque spaces, where inner workings of algorithms are not cognitively accessible to an operator, or operator controls are "abstract" with weak links to human cognitive processing. Within opaque spaces, MrS could attain different levels of opaqueness. So this space has been further classified into two categories – partially and fully opaque spaces.

As discussed earlier, Ron Arkin [1], allowed operators to change the value of wanderlust, the magnitude of random deviations from a planned path the robot was allowed. Using wanderlust in place of "magnitude of random deviations…," allows MrS controllers to associate behavior with a cognitively well-defined

operator. Similar cognitively well-defined operators are used by Lynn Parker [14]. She allowed the operator to adjust the twin tropisms impatience and acquiescence to affect a robot's willingness to persist or abandon a role. [8], by contrast, provided a steering mechanism that allowed the operator to directly manipulate the planned path.



In our taxonomy, [8] would be considered transparent while [1, 14] are categorized as systems with translucent algorithm steering, or simply translucent space.

E. Roth [15], asked her operator controllers to adjust weights through trial and error to find a plan containing a desired target. In this case, the operator controls changes in algorithm behavior but they are cognitively ill-defined. “Adjust weights through trial and error…,” is an abstract control with weak link to human cognitive processing. [15, 17] are therefore categorized into partially opaque space.

Considering the difficulties involved in achieving an autonomous MrS and the need for deployed systems to accept human input, a mixed-initiative approach is necessary. However, there exist large pools of MrS, where inner workings of algorithms are not cognitively accessible at all to an operator. Such cognitive inaccessibility results in high opacity and reduced performance. Chih-Han Yu [20] et al, for example, reported in their experiments using multirobot POMDPs that with increasing complexity of an office environment, controllers (i.e. humans) found it very difficult to observe or follow the optimal policy being executed by their robots. Similar full opacity can be seen in market-based optimization MrS. [6], for example, allotted leaders to subgroups to enhance market-based multirobot coordination. This introduction of leaders among subgroups resulted in improved performance but made it even more difficult for observers to follow. [20, 5, 6] are qualified members of fully opaque space.

Discussion

Classifying algorithms by degree of opacity is one potentially fruitful way to proceed because it orders coordination algorithms in terms of human difficulty. It is easy to imagine other classification schemes, however, that might be equally fruitful. The distinction between centralized and distributed control is lost in our current scheme but almost certainly plays an important role in determining effective forms of re-engineering. The more general problems of controlling biologically inspired systems with emergent behavior or affecting behavior through complex adjustments to teamwork algorithm parameters as attempted in [9] are not yet addressed within our evolving framework.

References

1. Endo, Y., MacKenzie, D., and Arkin, R. (2004). Usability evaluation of high-level user assistance for robot mission specification, IEEE Transactions on Systems, Man, and Cybernetics- Part C, 34(2), 168-180.
2. Beard, R., McLain, T., Nelson, D., Kingston, D., "Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs," IEEE Proceedings: Special Issue on Multi-Robot Systems, 2006.
3. J. W. Crandall, M. A. Goodrich, D. R. Olsen, and C. W. Nielsen. Validating human-robot interaction schemes in multitasking environments. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 35(4):438–449, 2005.
4. Cummings, M. and Guerlain, S. An interactive decision support tool for real-time in-flight replanning of autonomous vehicles, AIAA Unmanned Unlimited Systems, Technologies, and Operations, 2004.
5. Dias, M. B., Zlot, R. M., Karla, N., Stenz, A. T. Market-based multi-robot coordination: a survey and analysis. Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-05-13, 2005.
6. Dias, M.B., Stentz, A. Opportunistic optimization for market-based multirobot control. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, 2002, 2714- 2720
7. B. Gerkey and M. Mataric. A formal framework for the study of task allocation in multi-robot systems. International Journal of Robotics Research, 23(9):939–954, 2004.
8. Lenox, T., Hahn, S., Lewis, M., Payne, T. and Sycara, K. (2000) Task characteristics and intelligent aiding. Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics, Oct 8-11, Nashville, TN, 1123-1127.
9. Lewis, M., Polvichai, J., Sycara, K., and Scerri, P. Scaling-up human control for large scale systems, In Cooke, N., Pringle, H., Pedersen, H., and Connor, O. (Eds.) Human Factors of Remotely Operated Vehicles, New York: Elsevier, 237-250, 2006.
10. McLain, T. and Beard, R. Coordination Variables, Coordination Functions, and Cooperative Timing Missions, AIAA Journal of Guidance, Control, and Dynamics, vol. 28, no. 1, pp. 150-161, January-February 2005.
11. Miller, C. Modeling human workload limitations on multiple UAV control, Proceedings of the Human Factors and Ergonomics Society 47th Annual Meeting, 526-527, 2004.
12. Miller, C., & Parasuraman, R. Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control. Human Factors, 49.57-75, 2007.
13. Papadimitriou, C. (1994). Computational Complexity. Addison Wesley, 1994.

14. Parker, L.E., ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation, IEEE Transactions on Robotics and Automation , 1998
15. Roth, E., Hanson, M., Hopkins, C., Mancuso, V., and Zacharias, G. “Human in the loop evaluation of a mixed-initiative system for planning and control of multiple UAV teams,” Proceedings of the Human Factors and Ergonomics Society 48th Annual Meeting, 280-284, 2004.
16. Scerri, P., Vincent, R., and Mailler, R. Coordination of Large-Scale Multiagent Systems, Birkhauser 2006.
17. Schwager, M., Slotine, J. J., Rus, D. Decentralized, Adaptive Control for Coverage with Networked Robots. Proceedings of IEEE International Conference on Robotics and Automation, 2007, 3289-3294
18. Sycara, K. and Lewis, M. (2004). Integrating intelligent agents into human teams. In E. Salas and S. Fiore (Eds..) Team Cognition: Understanding the Factors that Drive Process and Performance, Washington, DC: American Psychological Association, 203-232.
19. Wang, J. and Lewis, M. Assessing coordination overhead in control of robot teams, Proceedings of 2007 IEEE International Conference on Systems, Man, and Cybernetics, 2645-2649, 2007.
20. Yu, C., Chuang, J., Gerkey, B., Gordon, G. J., Ng, A. Open-loop plans in multi-robot POMDPs. Technical Report, Stanford University, 2005

Mixed-Initiative Negotiation: Facilitating Useful Interaction Between Agent/Owner Pairs

Pauline M. Berry, Thierry Donneau-Golencer, Melinda Gervasio,
Bart Peintner, and Neil Yorke-Smith

Artificial Intelligence Center, SRI International, Menlo Park, CA 94025 USA
{berry,donneau,gervasio,peintner,nysmith}@ai.sri.com

Abstract. A mixed-initiative agent for personal time management interacts not only with its human owner but also with other agents and humans that share or depend on the same time commitments. The assistive capabilities of such an agent include the ability to provide information and context for its owner, negotiate on behalf of its owner, and understand when autonomous action is possible, preferred, or expedient. It must operate without losing the trust of its owner or negotiating partners. Since time management is intensely personal, each such human-agent pair will evolve its own characteristics and working practices. This position paper considers how an existing mixed-initiative adaptive time management agent, PTIME, can be extended to the multiagent negotiation setting. We discuss opportunities for facilitating personalized mixed-initiative negotiation protocols and adjustable autonomy through demonstration, instruction, and advice. In addition, we explore user interaction considerations including the provision of explanation to build trust by enabling the owner to understand and correct agent decisions and suggestions.

Key words: mixed-initiative, time management, negotiation, explanation, suggestion, trust

1 Introduction

Scheduling meetings in an office environment commonly depends on email-based negotiation. Typically, a series of messages between a meeting organizer and potential participants broadcast the intent to have a meeting, possible time windows for the meeting, restrictions on the windows, counter-proposals, and eventually agreement on a time, duration, location, and participants. While this method can produce the desired result, it comes at great cost to potential participants: in addition to the time spent on the viewing their calendars and responding to email, productivity is lost through the context switch required for each round of the negotiations. These costs motivate the need for automated assistance in scheduling.

A number of fully or semi-automated scheduling systems have been developed [1–3] but they have largely aimed to bypass the human negotiation as-

pects of meeting scheduling and to aim for full automation. The systems therefore do not fit within the organization's socially accepted (though commonly non-formalized) workflows regarding meeting scheduling, and so suffer from low adoption rates.

We have seen this lack of adoption with our own semi-automated scheduling agent, PTIME [4]. Each user has his own PTIME agent that assists in scheduling his own calendar and coordinating his calendar with others. The PTIME agent adapts to its user becoming personalized to his needs and preferences over time. Our aim is for the agent to work within users' typical scheduling workflows, rather than trying to convince users to switch to a different paradigm of meeting scheduling. In particular, we aim to automate aspects of an organization's current scheduling workflows and predict users' responses to decisions or requests for information in order to make suggestions to them or anticipate their responses. Eventually, the user may gain enough trust in his agent to allow it to make some decisions on his behalf. The larger goal is to reduce the time and number of context switches needed to organize or become an attendee in a meeting.

This time management domain leads us to address several interesting issues at the intersection of adjustable autonomy and multiagent negotiation. The combination of these two problems is less well studied in the literature, and we believe that addressing them together may produce synergistic opportunities and more intuitive agent behavior. This position paper discusses our initial thoughts on how a user and software agent can interact in the context of meeting scheduling and multiagent negotiation, and emphasizes the key contribution of user-agent dialogue to adjustable autonomy.

2 Mixed Initiative Negotiation and Personal Time Management

Artificial intelligence techniques can solve the multi-participant, distributed or centralized meeting scheduling problem, supposing sufficient information, However, existing fully or semi-automated scheduling systems fail to address the personal nature of the domain [1, 2]. The process of negotiating meeting times, the tools employed, and the preferences over events all exhibit considerable variation between individuals. For example, the best solution for an over-constrained meeting request (i.e., where not all criteria can be fulfilled simultaneously) depends on the individual: one person prefers to reschedule an existing meeting, another prefers times outside the specified window, while still another prefers to omit a participant. Hence, for a system that goes beyond standard calendaring functionality and addresses the scheduling problem to be of value, it must facilitate negotiation and embrace personalization.

Mixed-initiative systems integrate human and automated reasoning to take advantage of their complementary reasoning styles and computational strengths. Taking a mixed-initiative methodology is a powerful paradigm for engaging humans in a software process. Previous work in this area explores mixed-initiative interactions between the agent and the user in terms of a balance between agent

autonomy and user control [5]. Strategies are developed for allowing the user to personalize the default behavior of an agent, to subdivide tasks between the user and the agent according to the criticality of decisions, and to allow the user to inspect the agent's behavior and correct it. More recent work has argued that mixed-initiative systems must exhibit both *adaptable* strategies to allow the user to personalize the agent behavior and *adaptive* strategies that use AI techniques to personalize agent behavior automatically [6].

This development leads to an ongoing dialogue between the agent and its owner. The agent, by using this dialogue to refine its knowledge, can over time become a progressively more capable and trustworthy, scheduling agent [7]. Eventually this may lead to an agent with a self-adaptive ability. Myers and Yorke-Smith [8] discusses theories of proactivity in single agent behaviors. Work on Electric Elves [9] explores the problem of user-agent pairs in an organizational setting. Human agents were assisted in a variety of organizational tasks including meeting planning. Scerri et al. [10] explores practical progress toward adjustable autonomy in multiagent environments composed of human/agent pairs.

2.1 The PTIME Agent

PTIME (*Personalized Time Management*) is an intelligent, personalized calendar management agent that helps users handle email meeting requests, reserve venues, and schedule events. The agent is designed to unobtrusively learn scheduling preferences, adapting to its user over time. The ability to learn is a key aspect in the evolving interaction between human and agent. Details of the PTIME scheduling process and preference learners can be found in [4].

The PTIME multiagent environment consists of agent/human pairs interacting collaboratively to solve complex event scheduling problems. Figure 1 represents a set of PTIME/owner pairs within the context of a meeting negotiation initiated by human owner *A* and assisted by his PTIME agent *Ea*. Participants (*B,C,D*) and their PTIME agents (*Eb, Ec, Ed*) can share availability information and solution preferences, and can rank options. Note that not all participants need have a PTIME agent for the negotiation to progress. At present, the PTIME user organizing the meeting decides which meeting option to select, taking into consideration other participants' generic scheduling preferences. The selected meeting option is presented to invitees for inclusion or otherwise in their calendars. In the simplest form of negotiation supported, the other participants besides the organizer may simply accept the meeting request or not. PTIME is being developed to support more elaborate forms of negotiation while keeping the user in the loop. The question remains of why, when, and how such an agent should interact with the user and how that interaction should evolve over time.

2.2 An Example Multiagent Negotiation Use Case

In the context of our time management domain, there are two main types of interaction within a negotiation.

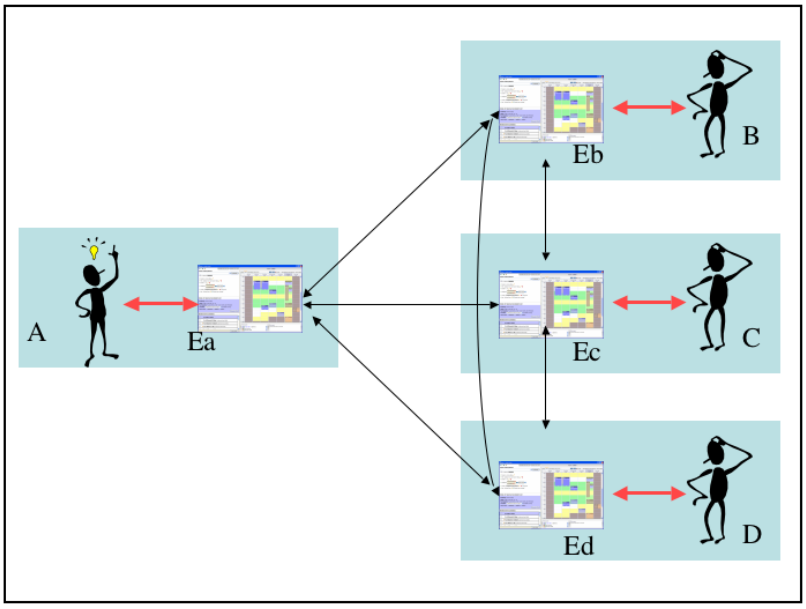


Fig. 1. Mixed Initiative Negotiation Use Case.

- A request for *information*: together, one agent/human pair can compose the request and the recieving agent/human pairs can answer. In each case the agent/human pair must evaluate the request and understand the context. The receiving agent/human pair can assess alternative responses and either answer or not. This form of request takes part during the negotiation process. An example use case is described in Figure 2.
- A request for a *decision*: in the case of meeting scheduling this typically takes the form of a request to accept or decline a suggested meeting time. Again, the agent/human pairs involved must evaluate the contents of the request, understand its context, assess alternatives, and respond. In this case the response may conclude the negotiation process or prolong it. A typical use case is described in Figure 3.

In a fully autonomous system the agent itself would manage these responses, but in a mixed initiative system it may require interaction between each agent and its human owner.

3 Facilitating Mixed-Initiative Negotiation in Agent Systems

We have argued that time management is a personal and sensitive aspect of people's day-to-day lives, and that an agent or assistant for this activity must be

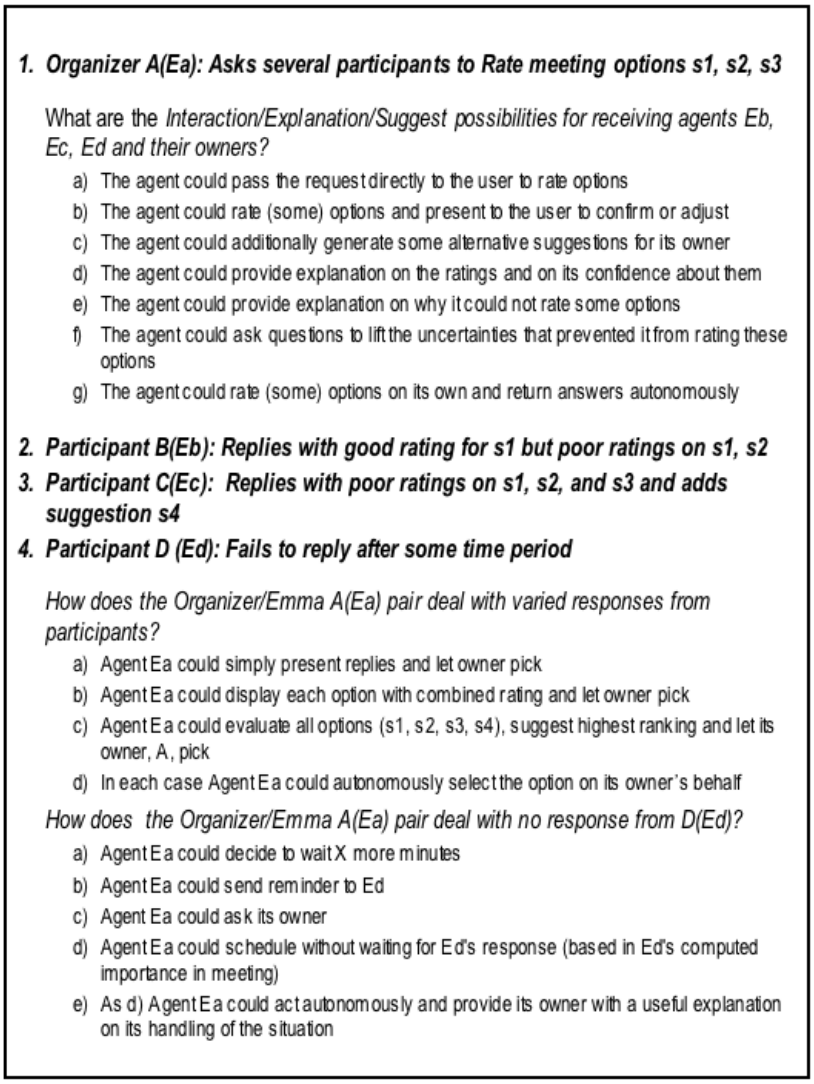


Fig. 2. Information-based Mixed Initiative Interaction in Negotiation Use Case.

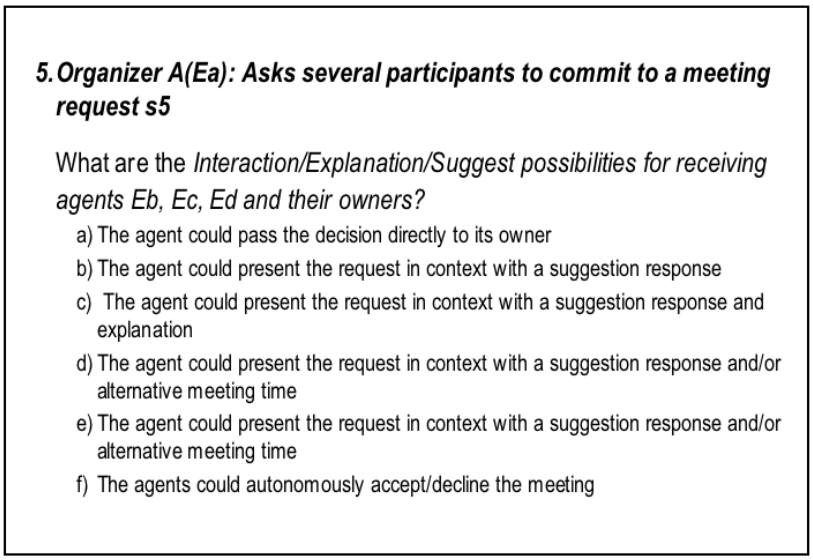


Fig. 3. Decision-based Mixed Initiative Interaction in Negotiation Use Case.

adaptable to its owner's needs. We have further argued that the agent must engage its owner in this process to both build trust and increase its ability to act for its owner over time. The thrust of our discussion is to explore useful interaction between the agent and its owner during the process of multiagent negotiation. We will discuss how the owner might inform the agent about acceptable behavior that is either desired by or helpful to the owner, and how in return the agent can assist its owner in understanding the context of the negotiation and possible actions available to, or autonomous actions taken by, the agent. The context is meeting scheduling and the interaction takes place when an agent/owner pair needs to share some information to satisfy the progression of negotiation, or the agent/owner pair must make a decision regarding the scheduling of a meeting. The mechanisms for interaction we will explore include observation, advice, suggestion, action, and explanation. These concepts are familiar to the field of intelligent user assistants [11, 12] and single agent-human adjustable autonomy [13] but can be applied to the problem of human-agent interaction in multiagent systems.

Interaction and the sharing of control between human/agent pairs are complicated by the multiagent environment. For example, what should be done if a participant in the negotiation fails to respond, or responds slowly? What options are available to the agent? Should it involve its owner and if so when? What happens when participants respond with unexpected or unhelpful answers?

Observation When a user makes a decision about what information to share or what action to take in a specific context, this is a demonstration to the agent about how that agent should behave under those exact conditions. Learning from these interactions is an unobtrusive form of learning that can impact an agent's behavior. It has some similarity to case-based reasoning, which has also been applied to learn scheduling preferences [14]. In PTIME we apply learning techniques to such interactions with the user to learn a model of user preferences, including scheduling preferences, location preferences, the importance of a person to a meeting or an event to a person. Each time a user makes a scheduling choice the agent will update its learned model of user preferences. This model is used to produce suggestions to the organizing user in the form of ranked options [4]. Similarly, it can be used to suggest a participant's response to a request for information or an accept/decline decision.

Our work in learning a model of user preferences has improved the ranking of meeting alternatives [4]. However, it is a more difficult task to derive rules for autonomous behavior from such models. Observations are noisy, the environment complex, and choices are often context dependent. Many research challenges remain in learning adjustable autonomy rules through observation. How can an agent break down its observations into chunks that are meaningful for a learner (for adjustable autonomy) and understandable to a user (for feedback). More likely, the observation must be combined with another type of interaction to be useful. For example,

- When an agent makes a mistake, the user can help it understand the problem and correct the learned rule (e.g., I declined this meeting not because of its time but because of its type).
- When an agent learns something, it can ask the user for confirmation that it is learning the correct rule.

Advice An agent should be able to accept its owner's advice and conform to organizational policies. Advice is defined as an enforceable, well-specified constraint on the performance or application of an action in a given situation. There can be hard advice, which can also be called *instruction*,and soft advice, which can be ignored by the agent and may define a preferred application of a constraint or action. [15] defines two types of policy: authorization and obligation. We extend this categorization to include preference:

- *Authorization* defines the actions that the agent is either permitted or forbidden to perform on a target.
- *Obligation* defines the actions that an agent must perform on a set of targets when an event occurs. Obligation actions are always triggered by events, since the agent must know when to perform the specified actions.
- *Preference* defines a ranking in the order, a rule of application, or a selection of an action under certain conditions.

Advice may be conflicting, it can be long-lived, and its relevance may decay over time. Advice can be used to influence the selection of procedures, negotiation

protocols, or choices and also to influence adjustable autonomy. The application of advice is central to both PTIME for influencing preference learning and in the future for learning adjustable autonomy strategies. In the latter case, some examples of advice in PTIME are to

- Always accept meeting requests from personname="Bob Smith" or personrole=MyManager.
- Respond automatically to a meeting time rating request when confidence in ratings is "high".
- Never schedule a meeting time after 4pm without my confirmation.

Several research challenges arise from the use of advice in adjustable autonomy. What language can be used for specifying advice? How should an agent apply advice, especially soft advice? Should an agent ever override, or ignore, advice? Finally, how should an agent handle conflicting advice?

Suggestions Suggestion or recommendation systems are often based on the ability to rate or rank options. In the same way PTIME can rate options presented to it and rank them to display to the user or share with another human/agent pair. In addition, PTIME can suggest an alternative option that might suit its owner better than those presented as part of an ongoing negotiation process. An interesting addition to simply rating options according to a static evaluation function is to use an evolving model of the owner's preferences. In PTIME this model is a reflection of previous interactions between the agent and its owner. Some sample suggestions may include

- In response to a *rate meeting times* request: PTIME suggests
  - 9am Tues (medium = 3 stars)
  - 11am Tues (low = 2 stars)
  - 3pm Tues (not suitable = 0 stars)
  - An alternative option 10am Tues (high = 4 stars)
- In response to a *schedule meeting* request: PTIME suggests *ACCEPT* with confidence level high

Particular challenges are created by the multiagent nature of the domain. Human/agent pairs are not necessarily reliable communicators. One agent may reply to a request immediately, having assumed responsibility for that action. Another may defer to its human owner and wait some undetermined time for a response. The initiating agent can use the mechanism of suggestions to present the user with alternative strategies to handle these particular cases.

Actions In a multiagent environment the agent may have some base level of autonomy and be capable of taking some actions, replying to another agent's request for information, updating a user's calendar, and so on. In a mixed initiative environment the human should be aware of these actions and understand their purpose and context. The research challenge is to enable the agent

to move from a base level of autonomy to the ability to act autonomously in a context-sensitive way. Even more challenging would be the ability to acquire this capability through natural interaction with the user. For example, if an agent repeatedly observes the user declining back-to-back meetings, the agent may begin to suggest this response based on its learned model. If the user accepts the suggestion repeatedly, the agent may in the future automatically decline such meetings and notify the user.

Explanation Explanation provides an agent with mechanisms for justifying suggestions or recommendations it may make, or actions it has taken, to a user [16]. A human when deciding to accept or reject a suggestion made by another human will consider the quality of previous recommendations from that person and how that person's interest aligns with his own. Where there is doubt, the human will ask *why*. In the same way, a human user will, over time, learn to trust (or not) the suggestions made by an agent and by extension the actions taken by an agent. While the relationship is developing, the user will want the agent to justify its suggestions and action. Thus, providing sensible and intelligible explanations is beneficial and necessary.

An important subcategory of explanation is the presentation of context to the user. This has several purposes: to help the user navigate the set of possible options, examine an option at multiple levels of abstraction, and annotate an option or change. Visual displays of options, *options at a glance*, are powerful methods of conveying context [17]. For example, the known status of a negotiation process can be displayed visually, or a set of options could be displayed within the known calendar information. An example of context in PTIME is shown in in Figure 4. Here options are displayed on a calendar also displaying shared calendar and preference information relevant to the current context.

In addition to presenting the user with a visual display of context and simple explanations of conflicts in terms of constraints violated, we would like to explore more specific forms of explanation in PTIME. For example,

- PTIME suggests 1pm on Tuesday as an alternative meeting time since everyone is available and PTIME has learned that the user prefers afternoon meetings
- PTIME suggests that the user decline this meeting request because it has learned he does not like back-to-back meetings with this host.

Explanations of a suggestion or action are influenced by the agent's reasoning process, the problem context, and the agent's knowledge of the user's preferences. In most scheduling systems it would be sufficient to explain a solution in terms of constraints violated and evaluation function used. However, if the agent is adaptive and concerned with learning how to be proactive then the challenge is to also explain a suggestion in terms of the learned model, i.e., "why do you think you are allowed to ...?" If an agent is going to reveal its learned preferences to the user this has implications. The user may wish to "correct" the agent. Thus, explanations must be dynamic to support user's follow-up questions.

Fig. 4. Context provided by options at a glance. The three options are presented in the context of the participants' schedules. Also, the system has information about two of the participants' preferences and the colors encode that information. The user can delve deeper into the context to look at the meeting details and view preferences individually.

4 Conclusions and Ongoing Work

This position paper sets out a discussion of the role of the dialogue between agent and human in the evolution of adjustable autonomy in the context of the time management domain. The discussion is based on our work on PTIME, a time management assistant that learns the user's preferences, and interacts with its owner to improve over time and earn the user's trust to act autonomously when authorized. As we expand the role of multiagent negotiation within PTIME we can benefit from existing work in mixed-initiative methodologies and distributed meeting negotiation to enhance the user's experience with PTIME. Our work can apply to any domain that has characteristics similar to our multiple human/agent pair environment and negotiation protocols.

Acknowledgments. We thank all the members of the CALO project, especially Khang Duong and Julie Weber. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or the Air Force Research Laboratory.

References

1. Higa, K., Shin, B., Sivakumar, V.: Meeting scheduling: An experimental investigation. In: Proc. of IEEE Intl. Conf. on Systems, Man, and Cybernetics. (1996) 2023–2028
2. Palen, L.: Social, individual and technological issues for groupware calendar systems. In: Proc. of CHI'99. (1999) 17–24
3. Tullio, J., Goecks, J., Mynatt, E., Nguyen, D.: Augmenting shared personal calendars. In: Proc. of UIST'02. (2002) 11–20
4. Berry, P.M., Gervasio, M., Peintner, B., Yorke-Smith, N.: Balancing the needs of personalization and reasoning in a user-centric scheduling assistant. Technical Note 561, Artificial Intelligence Center, SRI International (2007)
5. Cesta, A., D'Aloisi, D., Brancaleon, R.: Considering the user in mixed-initiative meeting management. In: Proc. of Second ERCIM Workshop on User Interfaces for All. (1996)
6. Bunt, A., Conati, C., McGrenere, J.: Insights from the design and evaluation of a mixed-initiative personalization facility. In: Working Notes of CHI'08 Workshop: Usable Artificial Intelligence. (2008)
7. Kozierok, R., Maes, P.: A learning interface agent for scheduling meetings. In: Proc. of IUI'93. (1993) 81–88
8. Myers, K., Yorke-Smith, N.: Proactive behavior of a personal assistive agent. In: AAMAS'07 Workshop on Metareasoning in Agent-Based Systems. (2007)
9. Chalupsky, H., Gil, Y., Knoblock, C., Lerman, K., Oh, J., Pynadath, D., Russ, T., Tambe, M.: Electric Elves: Applying agent technology to support human organizations. In: Proc. of IJCAI'01. (2001)
10. Scerri, P., Pynadath, D., Tambe, M.: Towards adjustable autonomy for the real world. J. AI Research **17** (2002) 171–228
11. Lamberti, P.M., Prager, J.M.: Advice-giving using reason: An intelligent assistant for interactive computing. In: Seventh IEEE Conference on Artificial Intelligence Applications. (1991) 428–434
12. Rich, E.: Users are individuals: Individualizing user models. Intl. J. Man-Machine Studies **18** (1983) 199–214
13. Isbell, C.L., Pierce, J.S.: An IP continuum for adaptive interface design. In: Proc. of HCI International. (2005)
14. Sycara, K., Zeng, D., Miyashita, K.: Using case-based reasoning to acquire user scheduling preferences that change over time. In: Proc. of IAAI'95. (1995)
15. Sloman, M.: Policy driven management for distributed systems. J. Network and Systems Managements **2**(4) (1994) 333–360
16. Glass, A., McGuinness, D.L., Wolverton, M.: Toward establishing trust in adaptive agents. In: Proc. of IUI'08. (2008) 227–236
17. Berry, P.M., Moffitt, M., Peintner, B., Yorke-Smith, N.: Design of a user-centric scheduling system for multifaceted real-world problems. In: Proc. of ICAPS'07. (2007)

Automated Collaboration among Communicating, Semiautonomous Vehicles

Dan Chevion[1], Ron Sivan[1], Onn Shehory[1], and Yuval Shimony[1]

[1] IBM Haifa Research Lab, Haifa, Israel 31905
{chevion, rsivan, shehory, yshimony}@il.ibm.com

Abstract. The general trend in combating traffic congestion and reducing accidents has shifted from paving more roads to making better use of existing infrastructure, often via technological improvements. Focusing on information technology, this paper combines several techniques in an innovative way to help communicating vehicles traverse conflict zones, such as merges and intersections, in an orderly fashion and at cruising speeds. This is done without taking away driver control of the vehicle, a property that reduces the automaker's liability exposure.

The first technique, known as tracking or Adaptive Cruise Control (ACC), can prevent a vehicle from getting too close to another vehicle. ACC is already available commercially for non-communicating vehicles. However, the reliability and benefit of tracking can be improved significantly once vehicles can communicate. The second technique introduced here, known as generalized tracking, allows communicating vehicles to track other vehicles not in their immediate vicinity, as long as those other vehicles are within communication range. This technique improves the ability of vehicles to coordinate road sharing among themselves. The third technique, known as traversal ordering, allows vehicles approaching a conflict zone to agree upon the order in which they are to traverse it. Combining these three techniques, we show how communicating vehicles can arrange themselves and cross conflict zones as quickly and safely as they do when traveling through a regular, non-conflict zone.

The fourth technique introduces acceleration into conflict zone traversal. From the perspective of flow, conflict zones can be viewed as a narrowing of the road, where the roads leading to it are capable of carrying more traffic than the conflict zone itself. This can, and often does, produce a backup that no amount of coordination can avoid. However, if vehicles are made to judiciously accelerate as they approach the conflict zone and decelerate once they have crossed it, flow through the conflict zone can be increased without affecting traffic outside of it. Safety is not jeopardized as the generalized tracking technique is applied as well.

We demonstrate the feasibility of these four techniques when applied to the case of a blocked lane via simulations.

Categories and Subject Descriptors. I.2.9, I.2.11.

General Terms. Algorithms, Design.

Keywords. Transportation, Automotive, V2V, V2I, Collaborative driving.

1 Introduction

Vehicular traffic is one of the more prominent woes of the modern industrialized world: accidents, congestion, and pollution are all major sources of concern. These problems are also interrelated—motor accidents are often the cause for congestion, and congested traffic pollutes much more than moving traffic does.

For many years, growth in traffic needs was addressed by increasing supply: wider roads, complex interchanges, etc. The return on such investments is declining, however, and newer solutions are being sought. Several recent solutions seek to control demand, such as designated roads or lanes (e.g., for public transportation) and prohibited zones (e.g., city centers), to name a few. Despite these efforts, vehicular transportation problems seem to keep growing. Much research is being conducted to find novel, long-term solutions that would permit better utilization of the existing infrastructure.

An alternative approach to alleviating the problem is to increase infrastructure utilization. Such an approach requires that the density of vehicles on the road be increased. Making the vehicles themselves smaller could help, but much more can be achieved by reducing the time and space margins vehicles usually keep between themselves for safety, since these margins take up much of the available road real estate. For example, a one-second distance between cars on the highway could account for 6 car lengths when moving at 100 km/hour; much time is wasted in deciding who goes next at merges and intersections and in other similar situations.

To increase infrastructure utilization without compromising safety, information technology solutions are required. Indeed, several solutions are being studied and some are already available commercially (e.g., Adaptive Cruise Control [2]). Yet most of these solutions do not assume communication, let alone collaboration, between vehicles. The study presented in this paper aims at solutions based on communication and collaboration among semiautonomous vehicles. We introduce several innovative techniques, some rather counter-intuitive, and combine them into a collaborative driving mechanism (Section 3.5). When applied to autonomous communicating vehicles, this mechanism and the underlying techniques solve well problems of the sort listed above.

In particular, the collaborative driving we present aims to facilitate fast and smooth passage of vehicles through segments of road where the straight-forward advance of vehicles is frustrated by other vehicles vying for the same space. Examples of such shared road stretches are intersections, lanes used for bypassing, merge areas, and lanes where vehicles wish to move at different speeds – all common situations in everyday traffic. Such a shared road stretch will henceforth be referred to as a conflict zone (CZ). CZs are the cause for many, and perhaps most traffic delays; vehicles must slow and often stop completely when negotiating passage through a CZ. They are also the site of 25% - 40% of all accidents [1] [5] .

Collaborative driving of the sort studied in this research relies on vehicle-to-vehicle and vehicle-to-infrastructure communications. Although this dependence raises issues of deployment, such systems may be practical even in the near future in controlled environments, where only communicating vehicles are permitted. Operational vehicles in a factory yard, or cabs for self-driving within an airport, are examples of such controlled environments.

Focusing on automated collaboration, this paper introduces several innovative techniques that, when combined, allow communicating vehicles to traverse CZs in an orderly fashion and at cruising speeds. The proposed solution places an emphasis on safety, reducing the chance of accidents.

Although the collaboration studied here is automated, its implementation is compatible with leaving a human driver ultimately in control of the vehicle. When driver actions interfere with the automated behavior, the system adjusts itself to deal with the new situation gracefully; the worst outcome would be some delay in traffic. We believe that this may make collaboration more acceptable both to drivers, who seek control, and manufacturers, who abhor the liability.

Section 2 lays the conceptual foundations for our collaborative driving mechanism. Section 3 delves into implementation details while Section 4 describes two possible scenarios to serve as examples. Section 5 describes previous work in the field, and Section 6 concludes.

2 Basic Concepts

In this section we introduce the basic concepts of four inter-vehicle coordination techniques. Implementation details of these techniques are presented in Section 3, and their combination into one mechanism is presented in Section 3.5.

2.1 Tracking

Tracking is used to maintain a safe distance between consecutive vehicles moving in the same direction in the same lane. This is not a new concept; it is already available commercially, and is described here only for the sake of thoroughness. A tracking vehicle behaves normally (i.e., in the same manner as contemporary vehicles do) when no other vehicle is ahead. Upon approaching another vehicle from behind, the vehicle is automatically constrained to move no faster than the vehicle it is approaching. This is considered a driving aid, helping drivers on highways and heavy traffic situations. It should be noted that the tracking capabilities now available in some high-end car models is sensor-based, using radars, lasers, or cameras mounted on the tracking vehicle to determine its relative distance from nearby vehicles. The present study, however, is concerned only with tracking based on inter-vehicle communications.

2.2 Generalized Tracking

As mentioned above, tracking may be achieved without communications, using vehicle-mounted sensors. Inter-vehicle communication, however, permits a vehicle to keep a *generalized distance* from any vehicle it can communicate with, whether in front, nearby or elsewhere, if only such generalized distance is defined. For example,

the generalized distance between two vehicles approaching the same CZ could be defined as the difference between their distances to that CZ; this boils down to the normal notion of distance when the two vehicles happen to travel along the same lane, but is different if they are not. The vehicle doing the tracking in such a situation is said to be *logically* tracking the other. This is similar to virtual vehicle mapping in [14] . From the driver's perspective, generalized tracking feels like ordinary tracking: the vehicle's acceleration is limited by the presence of some vehicle ahead. There could be cases, however, where the driver cannot see or is not aware of which of the other vehicles her own vehicle is tracking.

Since any change in the state of a vehicle can quickly be made known to all vehicles that are (physically or logically) behind it, even an abrupt stop is broadcast early enough to give affected vehicles enough time to slow down and avoid collision. The safety of the system is therefore unrelated to the speed with which participating vehicles are moving; their responses are not different from what they would have been had all vehicles been physically tracking. Any vehicle beginning to brake notifies all those behind it, which begin braking concurrently as a result. Due to the communication links, the fact that the row is only logical makes no difference.

2.3 Traversal Order

Currently, the standard way to allow vehicles through CZs is by time-division: vehicles take turns and cross the CZ one at a time. This sequencing could be imposed by rules of traffic (such as right of way) or by an external timing device (such as a street light). In any event, the essential aspect of these various solutions is the imposition of a sequence, or *traversal order* (henceforth TO) by which vehicles pass the CZ one after the other. This traversal order is currently figured out on-the-fly by the drivers, through watching the traffic and the relevant road signs. This procedure suffers from several flaws:

1. All drivers must decide on the same traversal order, or they might run into each other.
2. Calculating the traversal order takes time, and drivers slow down in the vicinity of CZs to give themselves enough time to figure out the correct traversal order.
3. When an external timing device is used, time is wasted when the right of way is given to a direction from which no traffic is coming. Moreover, timing devices often allow for some idle time when switching directions (a green light is given in one direction only some time after the red was turned on for all conflicting directions), adding to the delay they incur.
4. Time and energy are wasted when vehicles slow down at a stop sign or a street light, just to accelerate back to their original speed when given the right of way.

The approach we adopt here is to figure out the traversal order *in advance*. Once the order is decided and is disseminated to the vehicles involved, both the risk of misunderstanding could be eliminated and the time wasted on the slowdown and decision making could be saved. Note that this is different from scheduling, since only the order of the vehicles is determined, not their arrival times; there is no "missing one's slot" in this scheme.

2.4 Speed-up

The last observation has to do with the math of traffic flow. Let the following definitions hold:

- Flow (*f*): the number of vehicles crossing a given point in a unit of time, e.g., one hour (h)
- Density (*d*): the number of vehicles per unit length, e.g., one kilometer (km)
- Speed (*s*): the distance vehicles traverse per unit of time, e.g., kilometers per hour (km/h)
- Inter-vehicle gap (*g*): the space between two consecutive vehicles traveling in the same direction in the same lane

Using the definitions above, the following two mathematical relations hold (see [11] for example):

1. Flow is proportional to both density and speed. In particular, on average:

$$f = d \times s$$

2. Density and inter-vehicle gap are inversely proportional. In mathematical terms, if the average vehicle length is *l*, then

$$g = \frac{1}{d} - l$$

From item 1, we see that for a given flow, there is a tradeoff between density and speed; in particular, density may be reduced if speed is increased without changing the flow. From item 2, we see that reducing density increases inter-vehicle gap. Taken together, this means that by increasing travel speed, inter-vehicle gap may be increased without affecting the overall flow of vehicles.

Raising the overall speed of traffic is both difficult and risky. However, here the change is localized to a small area. If each vehicle in that area accelerates and then decelerates back to its original speed, the rates at which traffic enters and leaves the area remain the same as when no speed-up is employed, and yet the gaps between vehicles at the center of that area are widened, and can be used to accommodate interleaving vehicle flows.

3 Implementation

3.1 Implementing Generalized Tracking

Generalized tracking is realized similarly to ordinary tracking, with three main differences:

1. Information regarding the position of the tracked vehicle is most likely obtained through vehicle-to-vehicle communication (V2V).
2. The identity of the tracked vehicle is not self-evident and must be supplied.
3. The method by which distance to the tracked vehicle is measured must also be defined.

This information is sufficient for logic aboard the tracking vehicle to calculate the generalized distance at any time, based on its knowledge of the positions of itself and the tracked vehicle. We describe below two scenarios in detail, including their

respective rules for measuring generalized distances (see Sections 4.1 and 4.2). In both cases, generalized tracking is employed only in the vicinity of CZs, where an area controller (see Section 3.4) is available to coordinate traffic and can provide this needed information.

3.2 Implementing Traversal Order

Traversal order lists (TOs) are maintained for each CZ and list all the vehicles in the vicinity that wish to cross it in the order decided (see Section 2.3). Note that the list does not determine any absolute arrival or departure times, only the relative sequence.

Every vehicle in a TO, other than the first, follows a specific vehicle in the list. Tracking that vehicle guarantees the following vehicle will get to the CZ after that lead vehicle and all the other vehicles ahead of it have crossed the CZ. The TO may therefore be stored in a distributed fashion as a single piece of information each vehicle needs to carry, namely, the identity of the vehicle it immediately succeeds in the TO. This reduces the problem to a generalized tracking issue (see Section 2.2). Indeed, the vehicle being followed need not be the one physically ahead of the tracking vehicle; it could be approaching the CZ in a different lane or from a different direction altogether.

In reality, a vehicle may need to cross several CZs to reach its desired destination. In a four-way intersection, for example, a west-bound vehicle may be required to pass two lanes, one north-bound and another south-bound, before clearing the intersection. This amounts to two CZs, along with two TOs, and hence two vehicles to track, one from each TO. Since one of them would be nearer (in the generalized distance sense) than the other, keeping a safe distance from the nearest of the two will guarantee a proper relation with the other one as well. This argument extends to any number of CZs. Although the relation between vehicles in different TOs may change over time, requiring that the tracking vehicle keep monitoring all TOs it is in, such changes are relatively infrequent since they involve only the relative speeds of vehicles moving logically in the same direction.

3.3 Implementing Speed-up

As described above (Section 2.4), speed-up is done locally. Each vehicle is expected to accelerate until it reaches some maximum speed, cruise at that speed past the CZ and then decelerate back to its previous speed. The difference between the actual vehicle speed and the speed it would have had without speed-up is the *speed increment*. The speed increment follows the shape depicted in Figure 1. Note that the increment is dependent only on the position relative to the CZ.

A vehicle enters a speed-up condition when it is notified of the *speed-up scheme*, which is the relationship between speed increment and vehicle position relative to the CZ. Parameters of the speed-up scheme include the following:

1. The road position where acceleration begins
2. Dynamic shape of acceleration (may be variable)
3. The maximum speed to be attained
4. The road position where deceleration begins
5. Deceleration rate

Again, since speed-up is affected only in the vicinity of CZs, an area controller is available to provide this information. Under speed-up conditions, each vehicle calculates the speed increment continuously and applies it to its current speed; the increment is added to the speed the vehicle would have under the same conditions without the increment. Actual vehicle behavior is modulated by tracking, so its actual speed will never exceed that of the tracked vehicle. However, as we shall see, if all vehicles follow the same speed-up scheme, the tracked vehicle speeds up by the same amount and slightly before the tracking one, so tracking does not get in the way. Moreover, this delay is responsible for widening the gap between the two vehicles, which is the point of speed-up.

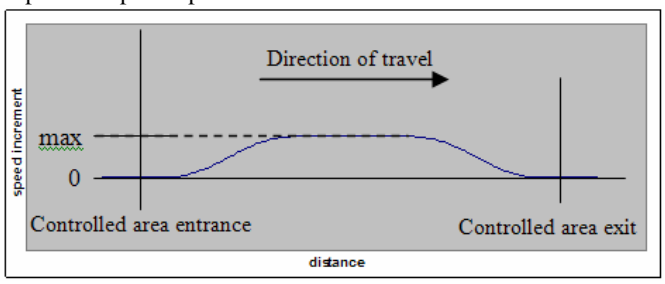


Figure 1: Speed-up scheme

3.4 The Collaborative Driving System

The system described pertains to an area containing a CZ, or possibly several related and closely-situated CZs. Such an area is equipped with a *control center* (CC), a server capable of coordinating traffic through the controlled area. Vehicles passing through the area need special capabilities to utilize the services of the CC and to streamline their passage. The properties of the vehicles and a CC are described below.

In this system, vehicles normally manage their advance based on the manipulation of their controls by their human operator, combined with information they glean via communications with other vehicles nearby. Vehicles inside the bounds of a controlled area of road also resort to the services of the relevant CC. Note that the CC does not violate vehicles' autonomy. It merely provides CZ-relevant information and coordination advice, yet vehicles communicate with one another in a peer-to-peer manner to gather the information needed and to collaborate accordingly.

Vehicle. A vehicle here refers to the common meaning of the term, augmented with hardware and software needed to perform the following tasks:

1. Communicate with other vehicles (V2V)
2. Communicate with CCs (V2I)
3. Generalized tracking
4. Speed-up

Control Center. The CC is a hardware device, stationary in most cases, which has jurisdiction over a specific stretch of road. It also has the hardware and software wherewithal required to perform its task, as described below. It is important to note that a controlled area may contain several CZs. The CC has the following responsibilities:

1. Set up the boundaries of the controlled area and notify vehicles that they have entered it.
2. Maintain a total order of all vehicles in the controlled area; maintain TOs for all CZs within the controlled area and notify vehicles of their placement in all TOs they participate in.
3. Determine the need for speed-up and dissemination of the speed-up scheme when it is in effect.

In the case of permanent CZs, such as those associated with an intersection, the CC could be installed where the street light is housed today, in addition or instead of the latter. A roadwork group would bring a CC along and install it in the vicinity of their work area for the duration of their work. An unexpected blockage, such as one due to a broken-down vehicle, could be controlled by equipment on an attendant emergency vehicle or even by the on-board intelligence of the broken-down vehicle itself.

3.5 The Mechanism

The mechanism presented below describes the behavior of a vehicle and a CC separately.

Vehicle Behaviors. Vehicles behave differently in non-controlled areas and controlled areas (i.e., areas under the jurisdiction of a CC). Furthermore, vehicle behavior in controlled areas requiring speed-up is different from that in controlled areas that do not.

Non-controlled area: In non-controlled areas, vehicles move in tracking mode (see Section 2.1).

Controlled area with no speed-up: Upon entering a controlled area, a vehicle switches to generalized tracking mode (see Section 2.2). The vehicle is informed that it has entered a controlled area, as well as of the identity of the vehicle it should track, through messages broadcast by the CC.

The new vehicle entering a controlled area need not be behind the vehicle it should be tracking now, but it may need to slow down to establish the proper tracking gap between itself and the tracked vehicle.

Controlled area with speed-up: In a control area employing speed-up, the CC conveys to an entering vehicle the parameters of the speed-up scheme (Section 3.3), in addition to the control information described in Section 3.2. The vehicle uses the scheme and its own position information to continuously calculate the momentary speed increment. It then does its best to implement the speed increment, subject to tracking constraints and the capabilities of its power train.

Control Center. The CC is responsible for maintaining the traversal order and to employ speed-up if the traffic rate warrants it.

Controlled area boundaries: The boundaries of the controlled area are set by the CC. The way they are determined could impact the efficiency of the system. The CC

could fix the boundaries at a certain distance from the CZ. This distance should be big enough to accommodate the ease of generalized tracking and consequent speed-up, if employed. CCs could also set the boundaries dynamically, starting out with narrow bounds, and expanding and shrinking the space taken up by the controlled area as momentary traffic rates change.

If the boundaries are set too narrowly, there might not be enough time for tracking and speed-up to relax. Boundaries that are set too widely, on the other hand, increase the chance of a slow-moving vehicle entering the controlled area and holding up faster-moving vehicles that have entered later but are now constrained to track it. It is therefore desirable to set the boundaries at their lowest practical distance. In that regard, dynamic boundary placement has an advantage.

One way to realize a dynamic boundary is through the placement of a priority line (PL) across all lanes leading to a common CZ at the same distance from the CZ. (Such lanes may or may not be adjacent; see detailed scenarios.) Initially, the PL is set at some minimal distance, dmin, but as vehicles pass the line, it is dynamically relocated to expand the controlled area in increments of ℓ, the average space taken up by a vehicle (its physical length plus the inter-vehicle gap at the speed it is moving). As vehicles cross the CZ and leave the area, the PL is moved backwards in decrements of ℓ, shrinking the area size. The PL is never set further than dmax. Thus, if there are n vehicles between the PL and the CZ, the PL would be at min(dmin + n ℓ, dmax). See Figure 2 for a schematic description.

Traversal order maintenance:

Vehicles crossing the PL are assigned monotonically decreasing priority values according to the order of their arrival. Note that a vehicle may cross the priority line due to its own motion, or due to the expansion relocation of the PL past the vehicle's current position. In either case, priorities are assigned according to the order in which vehicles and the PL meet. When two vehicles register within a short enough time period as to be indistinguishable, one of them is chosen to be first at random.

As an outcome, the CC can order all vehicles in the order of their arrival, which is assumed to reasonably correlate to the order at which they would have arrived at the CZ (to prevent the scheme from being unduly unfair). Emergency vehicles could enjoy preferential treatment.

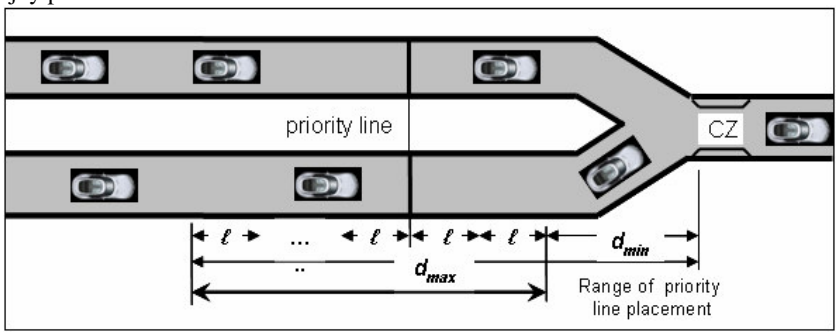


Figure 2: Placement of the PL where two lanes lead to a CZ. (PL is at dmin+ 2ℓ since two vehicles are between it and the CZ.)

This order is maintained within a TO. The CC notifies each vehicle, as it is placed in a TO, which TO it is in and which vehicle precedes it in the order; that is the vehicle it should track using the generalized tracking approach.

Speed-up management: As described earlier in this section, the position of the PL represents the number of vehicles contained between it and the CZ. The speed with which the PL itself moves represents the difference between the vehicles' arrival rate and the rate at which they pass the CZ. In the discrete case, the time between priority line repositioning is inversely proportional to the difference between the incoming and outgoing rates: the shorter the time, the faster vehicles ought to clear the controlled area.

Practically, speedup is triggered when the priority line reaches the distance of (dmin + dmax) / 2. The time at which that distance is reached is recorded as, say, t0. If the priority line is moved upstream at a later time, say t1, the speed of all vehicles that have crossed the priority line is increased by up to $\ell/(t1 - t0)$. Speed increment may be raised if PL movements accelerate, but is never reduced: it is reset back to 0 if and when the space between the PL and the CZ is cleared of vehicles.

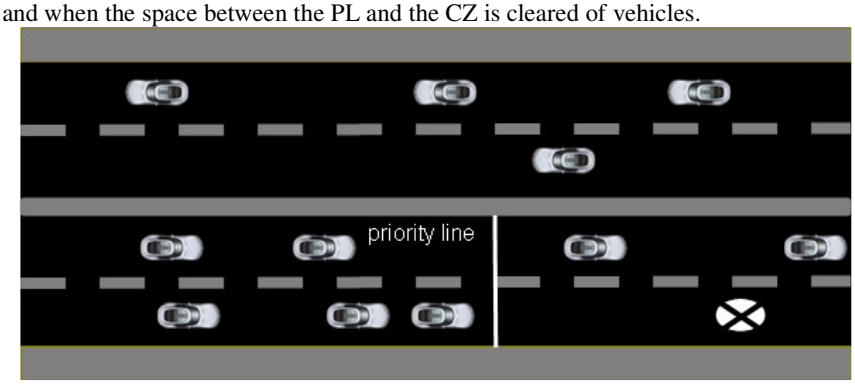


Figure 3: Vehicles approaching a blocked vehicle in a multi-lane highway.

4 Detailed Scenarios

4.1 A Blocked Lane in a Multi-Lane Highway

The system above may be applied to a multi-lane highway where a lane is blocked, due perhaps to a broken-down vehicle or road work. The associated CZ is the space in the unblocked lane next to the blockage, through which traffic arriving on the blocked lane will eventually have to travel. (If there are more than one unblocked lanes, each one may have a CZ.) The controlled area extends from the blockage to the PL, which may be dynamically set as described in Section 3.4 (see Figure 3).

Vehicles that have crossed the PL are within the controlled area and are assigned priorities. The metrics for the generalized tracking in this case is simply the distance to the blockage. Prioritized vehicles begin following all vehicles with a higher priority than their own, regardless of their respective lane position. (Recall that priorities are assigned in decreasing order.) This causes vehicles to arrange themselves in a single file even if they are different lanes (see Figure 4).

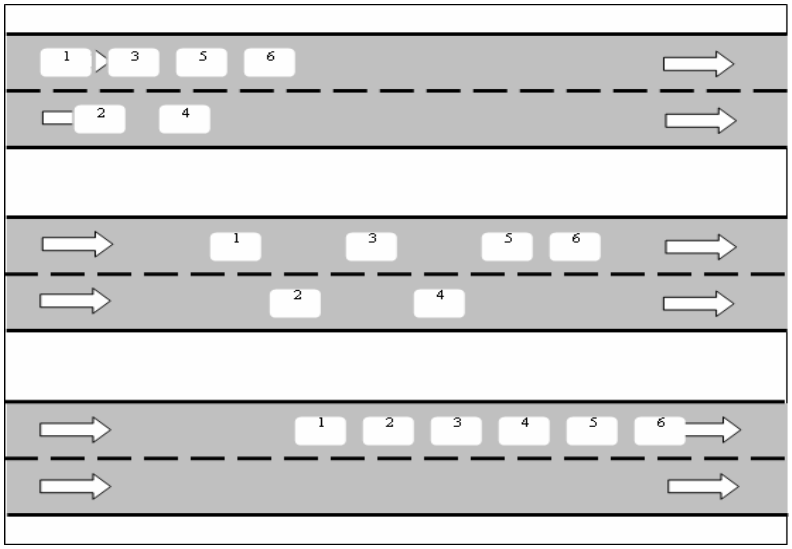


Figure 4: Tracking and merging by priority; the numbers indicate relative priority. Top - initial state; middle - after tracking was applied; bottom - traffic merged into one lane.

Once that state has been reached, any vehicle in the blocked lane (the bottom one in Figure 4) may move to the unblocked lane with confidence, knowing that all vehicles in the target lane with lower priority are already tracking its position, and as a result maintain a gap for it. (That is, the gap between vehicles 2 and 4 in the middle part of Figure 4 is maintained regardless of which lane vehicle 3 is in.) Lane switching is the responsibility of the driver – tracking modulates only vehicle speed, not steering – and must be completed after the vehicle crosses the line and before it reaches the obstacle. Vehicles which fail to move to the open lane in time get blocked, and have to wait, as all vehicles currently must, until traffic in the other lane has subsided.

Sufficiently sparse traffic could merge into the single open lane without any slowdown. At higher rates, however, a slowdown would result: while the first vehicle in a platoon maintains its previous speed, vehicles behind it must slow in order to generate the required inter-vehicle gaps. (E.g., in Figure 4, in the period between the situation depicted at the top and that in the middle, vehicle 6 has moved a longer distance than vehicle 1. Consequently, vehicle 1 must have moved more slowly than vehicle 6, which would not have necessarily been the case without tracking.)

To avoid such slowdowns, vehicles that have passed the priority line are made to accelerate, increasing lane capacity to the level necessary to accommodate the traffic at hand. The speed-up is controlled by the system, and the driver has only limited control over it. Although not necessarily implemented that way, one may think of the added speed as imparted by movement of the pavement itself, as in a conveyor belt, adding a constant speed to all vehicles that are (virtually) on it. The speed of the conveyor belt is determined by the amount of traffic that it needs to accommodate. At most, its speed could reach the average traffic speed, effectively doubling the capacity

of the single lane, and thus matching the throughput of both lanes together (at the given average traffic speed).

Once a vehicle has passed the obstacle, it decelerates back to its original speed, since both lanes are now available to carry the traffic and no speedup is needed any more. The effect of the speedup is therefore limited to a small area in the vicinity of the blockage and has no systemic effect on traffic as a whole.

The performance of this mechanism in this scenario was tested using a simulator built using the AnyLogic [3] agent-based simulation platform. The program is a straight-forward implementation of the mechanism described above over a stretch of a two-lane highway. Each vehicle is represented by an agent having one of three states:

1. Free-flow, in which the vehicle travels without any regard to vehicles around it.
2. Tracking, in which the vehicle follows behind another vehicle at that vehicle's speed
3. Obstacle, in which the vehicle is stationary

Each vehicle has properties of position and speed, as well as the identity of the vehicle it follows when tracking. Property values are chosen at random from a realistic range for highway traffic. Vehicles in tracking mode can switch lanes, if there is room in the adjacent lane. On-screen controls allow switching the obstacle on or off, the setting of the arrival rate, and selection of the algorithm to use: none, generalized tracking, and generalized tracking with speed up (see Figure 5).

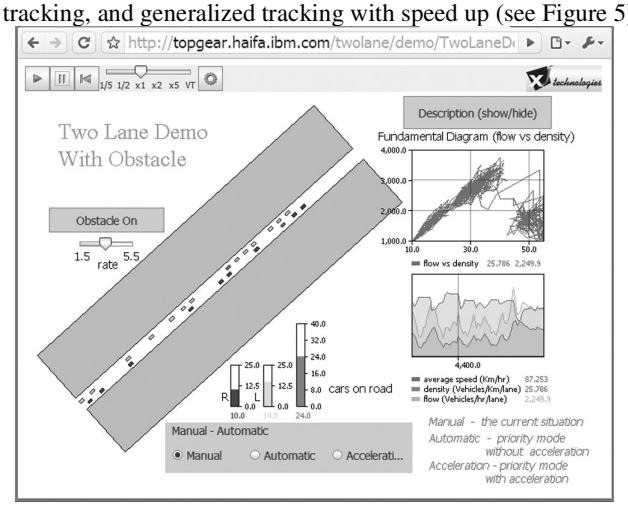


Figure 5: Snapshot from the collaborative driving simulator

At traffic rates of over 1500 vehicles/hour/lane, a traffic jam quickly forms when an obstacle blocks one lane, but it dissipates if generalized tracking is turned on. Tracking can handle traffic rates up to about 2200 vehicles/hour/lane. If speed-up is also employed, traffic rates of even up to 3000 vehicles/hour/lane do not form a jam.

A flow vs. density diagram for the simulated traffic, which the simulator maintains, exhibits the characteristic fundamental diagram behavior. An applet simulating this scheme under various traffic conditions is available at [4] .

4.2 Intersection Where No Turns Are Allowed

Another example is that of an intersection where no turns are allowed; all vehicles are constrained to cross the intersection going straight only. There could be any number of lanes in each direction, however.

To simplify the exposition, an intersection consisting of two one-way intersecting lanes, as in Figure 6, is described first; the solution is then expanded to any number of lanes. The two lanes form exactly one CZ: the area common to the two lanes. The bounds of the controlled area are set by a PL, which is placed, possibly dynamically, on both incoming lanes at equal distance from the CZ.

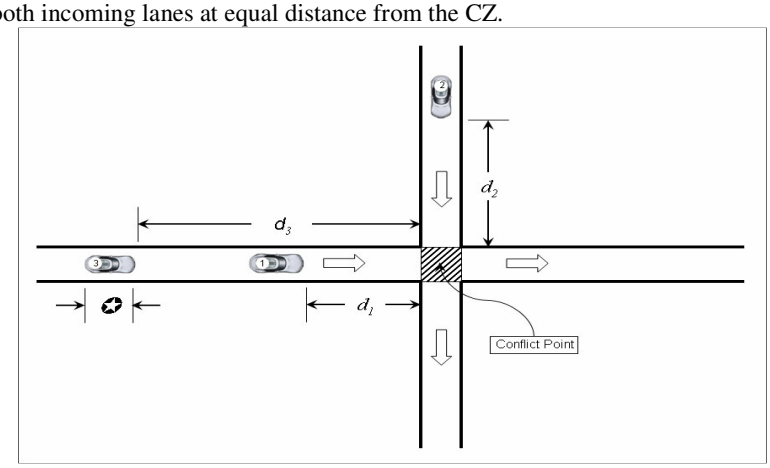


Figure 6: An intersection with a single CZ, defining the metrics for generalized tracking.

For the purpose of generalized tracking, the metrics are defined as follows:

1. The distance between two vehicles traveling in the same lane is measured in the normal way—the distance one vehicle must travel to touch the end of the other.

2. The distance between two vehicles approaching the intersection from different roads is defined as the difference between their respective distances to the part of the CZ that is closest to each.

For example, in Figure 6, the distance between vehicles 3 and vehicle 1, which is physically ahead of vehicle 3, is given by

$$d_{31} = d_3 - d_1 - \ell$$

where ℓ is the average length of a vehicle. The distance between vehicle 2 and vehicle 1, which are approaching the CZ from different directions, is given by

$$d_{21} = d_2 - d_1 - \ell - w$$

where w is the average width of a vehicle. The added padding is needed because the distance a vehicle travels from the point it enters the CZ to the point it clears is the sum of the vehicle's length and the width of the CZ, which is about the width of a vehicle. If all vehicles are moving at the same speed, that should also be the gap between vehicles moving perpendicular to that vehicle (see Figure 7).

A likely TO for the situation depicted in Figure 6 is (1, 2, 3).

The same idea applies to intersections in general. The area common to any intersecting lanes is considered a CZ, and a TO is maintained for each (see Figure 8). Each vehicle computes which CZs its path must cross, and which vehicle is just ahead of itself in each of the associated TOs. Every such TO contributes a vehicle it must follow, and each of these vehicles imposes some distance from the intersection it must keep at any moment. The maximum of these distances satisfies the tracking requirements of all of them, and that is the distance the vehicle should maintain.

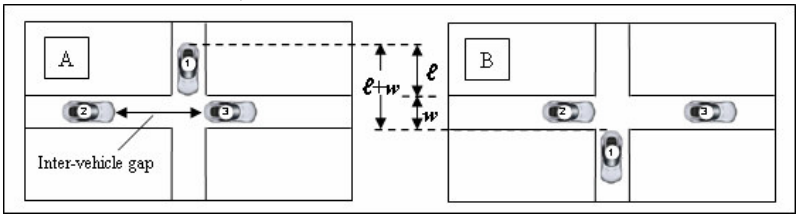


Figure 7: A vehicle crossing an intersecting traffic stream needs a gap as wide as its length plus its width. The distance vehicle 1 travels since entering the intersection (A) until it clears it (B) is ℓ + w; assuming 2 and 3 move at the same speed, their inter-vehicle gap must be just as large.

For example, vehicle 2 in Figure 8 must cross CZs B1 and B2 to get to the other side of the intersection. The TO for CZ B1 contains vehicle 4, and that of CZ B2 contains vehicles 1 and 3. Vehicles 1 and 4 should pass the intersection before 2, so 2 must follow them both. Although the speeds in Figure 8 are not known, it stands to reason that vehicle 2 follows vehicle 4, and that will take care of vehicle 1 as well, since vehicle 1 is so much closer to its CZ than vehicle 4 is.

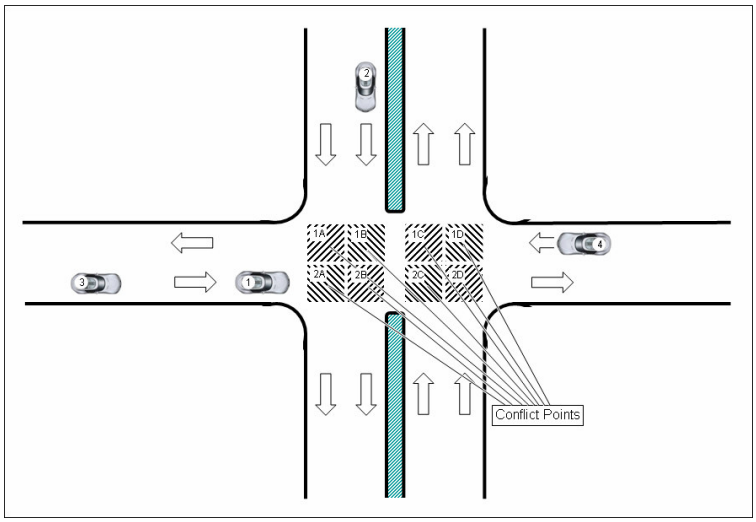


Figure 8: An intersection with several CZs.

5 Related Work

Collaborative driving is seen as a promising solution to mounting traffic problems. A collaborative scheme for traversing intersections with a setup similar to the one used in this paper is described in [8] [9] ,[10] [12] Dresner and Stone, as well as Kolodko and Vlacic, describe the intersection as being divided into vehicle-size slots in time and space, and each approaching vehicle can request and obtain a rendezvous with such a slot. If all such rendezvous are kept, safe passage is guaranteed without the need to slow down, but a vehicle that misses its turn must stop, delaying itself and the vehicles behind it. Our solution avoids such delays. On the other hand, the other scheme supports turns, whereas ours, at this stage, does not. The other scheme can be integrated with a standard street signal to accommodate some non-communicating vehicles, although performance suffers if there are too many of such vehicles. That scheme further requires that vehicles' behavior in the vicinity of the intersection be governed completely by the CC. Our scheme does not impose such a restriction, and therefore can be more gracefully adopted.

A scheme that requires no control center is described in [13] There, a scheme similar to the generalized tracking is used to serialize *vehicle pairs* that can occupy the intersection simultaneously without interference (for instance, if they are moving in opposite directions and both intend to turn right). Their trajectory is then automatically controlled to realize the best traversal plan. In that scheme, too, the driver has no control of the vehicle while it crosses the intersection, while in our solution the driver retains control.

We are not aware of other collaborative driving schemes that specifically address intersection and general CZ crossing. However, several multi-agent and distributed AI theories could help solve this problem. For instance, the generalized partial global planning framework (GPGP) [6] , a general scheme for distributed coordinated planning, can be used. Indeed, GPGP has been applied to the distributed vehicle monitoring testbed (DVMT) [7] That application exhibits similarities to the generalized tracking presented here; however, it does not refer to the specific problem of CZ crossing. It further assumes full autonomy of vehicles, whereas we allow driver intervention.

6 Conclusion

In this study, we have introduced an innovative inter-vehicle collaboration mechanism. When implemented, it allows (semi-) autonomous vehicles to increase road infrastructure utilization with virtually no effect on safety. As we show, our mechanism lets vehicles traverse road conflict zones such as blocked lanes and intersections with significant time and energy savings, avoiding decelerating and accelerating which would otherwise be required. In a counter-intuitive manner, we advocate that vehicles should accelerate into a busy road intersection. This is shown to be advantageous and introduces no additional risk when complementary techniques are implemented.

As shown in simulations, the suggested techniques are promising. Future work calls for extensions to support turns, simulations of additional scenarios, and eventually field studies. With such studies, we are confident that techniques of the sort presented here should become part of future driving reality.

7 Acknowledgements

The authors wish to thank Stuart Feldman, Larry Lieberman, Peter Robisson, and Dov Ramm for their invaluable support and advice.

8 References

[1] A Quick Look at Intersection Crashes in Canada. Road Safety and Motor Vehicle Regulation Directorate, Transports Canada, http://www.tc.gc.ca/roadsafety/tp2436/rs200806/menu.htm, (2008)
[2] Adaptive Cruise Control, http://www.mobileye.com/default.asp?PageID=328.
[3] AnyLogic 5. XJ Technologies, http://www.xjtek.com.
[4] Blocked Lane Simulation, http://topgear.haifa.ibm.com/ twolane/demo/TwoLaneDemoApplet.html. A narrated introduction, http://topgear.haifa.ibm.com/ twolaneMovieEng/take6Web.swf.
[5] Crash Rates at Intersections. Kentucky Transportation Center Research Report KTC-03-21/SPR258-03-2I, http://www.ktc.uky.edu/Reports/KTC_03_21_SPR258_03_2I.pdf, (2003)
[6] Decker, K., Lesser, V.: Generalizing the Partial Global Planning Algorithm. Intelligent and Cooperative Information Systems, vol. 1, pp. 319-346 (1992)
[7] Decker, K., Humphrey, M., Lesser, V.: Experimenting with Control in the DVMT. In Proc. *Third Annual AAAI Workshop on Blackboard Systems*, Detroit (1989)
[8] Dresner, K., Stone, P.: Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. In Proc. AAMAS'04, pp. 530-537, New York, USA (2004)
[9] Dresner, K., Stone, P.: Multiagent Traffic Management: An Improved Intersection Control Mechanism. In Proc. AAMAS'05, pp. 471-477, Utrecht, The Netherlands (2005)
[10] Dresner, K., Stone, P.: Sharing the Road: Autonomous Vehicles Meet Human Drivers. In Proc. IJCAI 07, Hyderabad, India (2007)
[11] Kerner, B. S.: The Physics of Traffic. Springer-Verlag (2004)
[12] Kolodko, J., Vlacic, L.: Cooperative Autonomous Driving at the Intelligent Control Systems Laboratory. *IEEE Intelligent Systems*, 18(4):8–11 (2003)
[13] Li, L., Wang, F.-Y.: Cooperative Driving at Blind Crossing Using Intervehicle Communication. IEEE Transactions on Vehicular Technology, Vol. 55, No. 6 (2006)
[14] Uno, A., Sakaguchi, T., Tsugawa, S.: A Merging Control Algorithm Based on Inter-Vehicle Communication. IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (1999)

Mixed-Initiative Cyber Security: Putting humans in the *right* loop

Jereme N. Haack[1], Glenn A. Fink[1], Wendy M. Maiden[1], David McKinnon[1],
and Errin W. Fulp[2],

[1] Pacific Northwest National Laboratory, Richland WA, USA
[2] Wake Forest University, Computer Science Department, Winston-Salem, NC, USA
{jereme.haack, glenn.fink, wendy.maiden, david.mckinnon}@pnl.gov, fulp@wfu.edu

Abstract. Organizations and their computer infrastructures have grown intertwined in complex relationships through mergers, acquisitions, reorganizations, and cooperative service delivery. Consequently, defensive actions and policy changes by one organization may have far-reaching negative consequences on the partner organizations. Human-centric and machine-centric approaches are insufficient for defending the security of today's increasingly complex computer infrastructures. The former are slow but highly adaptive, while the latter are fast but highly specialized. We believe the solution lies in mixed-initiative defenses combining the complementary qualities of both human- and machine-based approaches. We describe the Cooperative Infrastructure Defense (CID), a new cyber-defense paradigm designed to unify complex-adaptive swarm intelligence, logical rational agents, and human insight. CID will enable cooperative defense of infrastructure through situational awareness using visualization, security policy dialogue between humans and agents, shared initiative in solving cyber problems, and a foundation for building trust between humans and agents within and between organizations.

Keywords: agents; security; mixed-initiative

1 Background

Computer infrastructures consist of multiple interdependent organizations that share computational resources in a formal or *ad hoc* arrangement. In critical infrastructures such as electrical power grids, organizations may be so interdependent that failure of some part of a single company could produce cascading failures with consequences on the local, national or international scale. Regardless of the degree of their interdependence, infrastructure members remain economically independent organizations. They have separately managed systems, few shared policies, differing business drivers, and proprietary information that cannot be shared without special legal instruments.

To manage the complexity, we factor these multi-organization cyber infrastructures into *enclaves*, a set of computer and network hardware and software that is owned by a single organization and administered under a unified policy by a single (possibly separate) organization. An enclave may be very small, consisting of one or two machines, or it may be very large, comprising numerous networks. Enclaves are the building blocks of infrastructures and are the largest unit over which a mixed-initiative system [1] can be fielded without crossing proprietary boundaries between organizations.

Accomplishing concerted cyber defensive action that spans organizational boundaries is difficult. Infrastructures have unique cyber security needs that cannot be adequately addressed by individual enterprise solutions, and the complicated relationships in infrastructures make it possible for defensive actions by one organization to affect the others adversely [2]. Both legal [3, 4] and practical issues require that the consequences of change be managed through coordination across organizations in a near-real-time manner.

Cyber adversaries, on the other hand, are not hindered by central coordination; they can rapidly and concertedly disrupt multi-organizational computer infrastructures. Therefore, enclave defenders need intelligent defenses with the autonomy to adapt in real-time to both internal and external threats. The real-time nature of threats on the Internet requires that humans not become a bottleneck, and the seriousness of proprietary boundaries requires that automated defenses strictly observe the policies of the cooperating

organizations. A mixed-initiative approach solves these problems by allowing not only shared control between humans and software agents but also shared initiative among human stakeholders across the infrastructure.

Current cyber defense systems involve humans at multiple levels, but people are often far down in the control structure, requiring them to make too many time-critical decisions. Information flow between humans is slow and frequently asynchronous. In a crisis, humans may be unable to cooperate because of culture, language, legal, proprietary, availability, or other obstacles. Such systems cannot adapt to the Internet speeds of cyber threats. Consequently, effective cyber defense requires a framework that simultaneously capitalizes on the adaptability of humans and the speed of machines. In other words, humans must be put in the *right* loop to maximize their effectiveness while preserving their legal responsibility for the actions of their autonomic systems [4].

2 Introduction

This paper presents a new mixed-initiative hierarchical framework of humans and agents that is well suited to protecting computational infrastructures. The framework, Cooperative Infrastructure Defense (CID), is designed to rapidly adapt to new cyber attacks via swarming software agents while enabling humans to supervise the system at an appropriate level. We interpose a hierarchy of rational software agents between the swarm and the human supervisors to provide a channel for system guidance and feedback. Using various kinds of rationality actually turns false positives into beneficial forms of positive feedback and improves system performance.

CID represents a revolutionary new way of looking at system control for cyber security. Traditionally, humans control the entire system assisted by automated tools and subsystems. In CID, agents share the decision-making power, handling most of the real-time portion autonomously but enabling human involvement at all levels. The human supervisor does not directly control the system rather humans exert supervisory influence sharing the initiative for action with their software agents. CID is designed to be a scalable, dynamic, and robust framework for securing increasingly complex computational infrastructures. CID makes humans an intrinsic part of the solution, engaging them without requiring them to directly control and enables diverse organizations within an infrastructure to cooperate in an adaptive cyber defense.

In our research, we have created simulations of the entire framework, prototypes of the user interface, and prototypes of the mobile sensor agents. Our initial prototype, built in an agent simulation framework, was demonstrated at the VizSec 2008 conference. We are currently implementing the framework in Java using the JADE agent framework (http://jade.tilab.com/) on a network of virtual machines.

The remainder of this paper is structured as follows. CID is described at a high level in Section 3. Section 4 describes examples of human-agent and agent-agent interaction in the CID framework. Section 5 discusses related work. The conclusion outlines possible directions for future research, summarizes the CID framework, and reviews the expected benefits.

3 System Overview

In CID humans and various types of software agents share the responsibilities of securing an infrastructure comprised of enclaves that belong to member organizations. Figure 1 shows how one human can supervise a multi-enclave system with a few enclave-level agents, a host-level agent at each machine or group of similar machines, and a large swarm of simple mobile agents. Our terminology is as follows:

- Humans function as *Supervisors*. They provide guidance to and receive feedback from one or more enclaves. They must to take action only when the lower-level agents encounter a problem that requires human involvement. Supervisors may take initiative as desired inspecting and guiding any element of the system. However, direct human *control* of the system is discouraged because such involvement would destroy its natural adaptive abilities.

- Enclave-level agents called *Sergeants*, which are each responsible for the security state of an entire enclave. Sergeants may make service agreements with Sergeants of other enclaves. Sergeants dialogue with humans to gain guidance for running the system according to business drivers and human security policies. Sergeants create and enforce executable policies for the entire enclave.

- Host-level agents called *Sentinels*, which are responsible for protecting and configuring a single host or a collection of similarly configured hosts such as a cluster or storage network. Sentinels interact with human supervisors only when they need clarification about how to classify ambiguous evidence from the swarm.

- Swarming agents, called *Sensors*, roam from machine to machine within their enclaves searching for problems and reporting to the appropriate Sentinel. Sensors are diverse; their classifiers are each uniquely derived from the set of known problem indicators. Sensors use stigmergic messages called *digital pheromone* [5] to communicate.

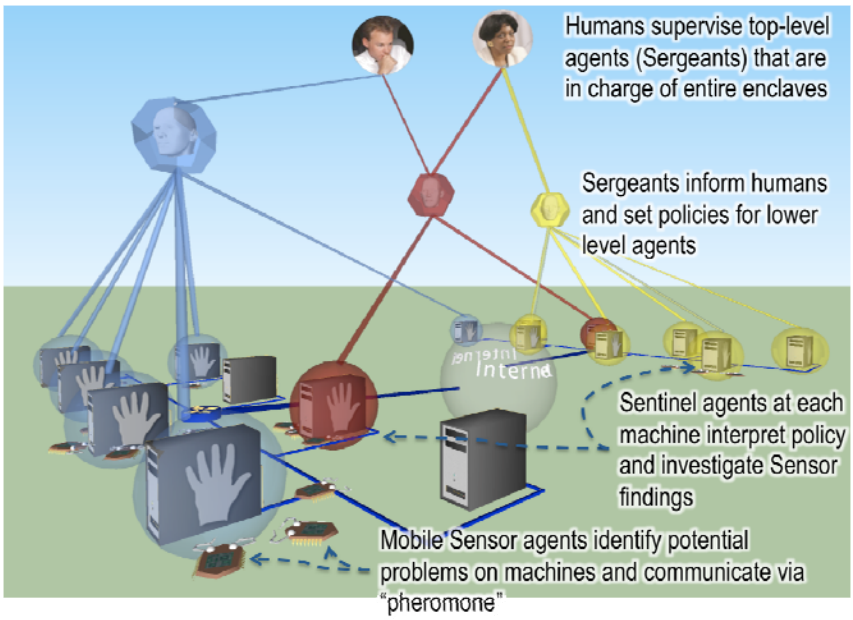


Figure 1: CID is a hierarchical framework of human supervisors, enclave-level rational agents, and swarming agents. A single human may supervise multiple enclaves via the agent hierarchy.

The concept of supervisors and agents of the CID framework operating within a hierarchical structure is supported by the research of Parunak [6], Smieja [7], and Selfridge [8], who each suggested hierarchical arrangements of heterogeneous agents. Interposing logic-based rational agents between the humans and the swarm provides a basis for communication, interaction, and shared initiative. The hierarchical arrangement gives humans a single point of influence that allows multiple points of effect. The following sections describe the roles of each actor in the CID and the relationships between actors are depicted in Figure 2.

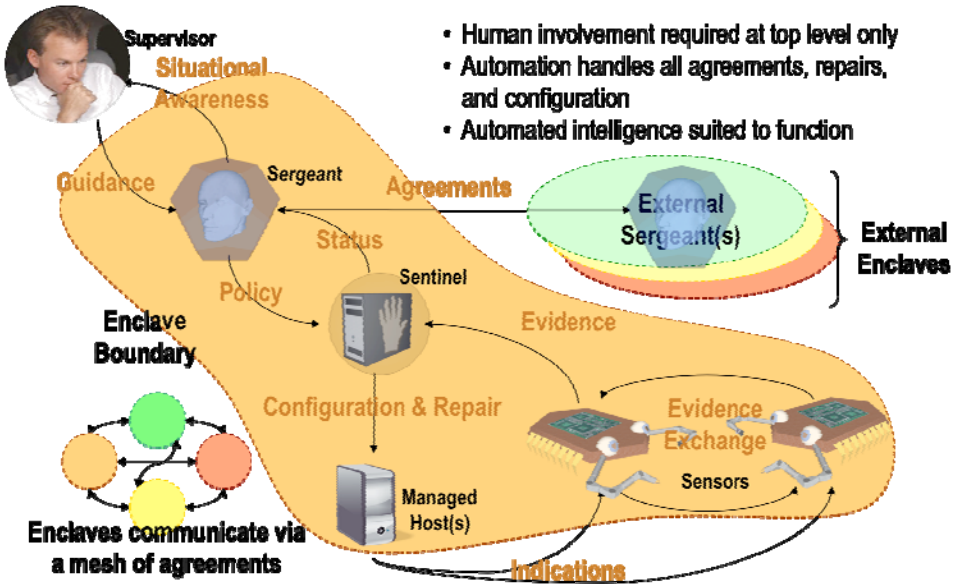


Figure 2: Cooperation among humans and agents in CID.

3.1 Supervisors

At the top layer of Figure 1 and Figure 2 are human supervisors who may direct one or more enclaves. Supervisors may belong to one or more interdependent organizations within the infrastructure, while the cyber assets in every enclave all belong to a single organization by definition. A Supervisor might also be a member of a regulatory organization or law enforcement agency and only monitor the equipment in the enclaves. Human supervisors translate business policy into guidance via natural-language and graphical controls for top-level agents called *Sergeants*.

3.2 Sergeants

Each enclave has a top-level agent called a Sergeant. In autonomic computing [9] terms, the Sergeant corresponds to the orchestrating autonomic manager. Sergeants provide situational awareness to their Supervisor, via an information visualization interface. Sergeants translate the guidance from the human Supervisor into actionable policy across all the machines within an enclave. We believe this will involve a natural-language dialogue that accepts human guidance and feeds back what the translation will be in terms of policy similar to IBM's SPARCLE (Server Privacy ARrchitecture and CapabiLity Enablement) policy workbench [10]. Sergeants will employ supervised learning algorithms so that interactions with them become more efficient over time. Sergeants are "heavy-weight" rational agents that make decisions based on logic. One possible implementation we have considered for Sergeants is Belief-Desire-Intention (BDI) logic [11]. The Sergeant presents the activities of lower-level agents to the human Supervisor and functions as an interface to influence system operation. Supervisors use Sergeants to enact environmental settings and policies that govern the general operation of the lower-level agents without controlling the lower-level agents directly. Sergeants provide a "geography" for the enclave that enables the swarm to operate properly.

Another major function of Sergeants is to broker agreements between CID enclaves on behalf of the Supervisors. To ensure that their actions are properly attributable, Sergeants must have a separate digital identification from the Supervisor they report to. Since they negotiate on behalf of humans, they may incur liability for their owning organization. Thus, there must be a mechanism to describe the types and degrees of authority the Supervisor has delegated to the Sergeant. Often, this authorization can be quantified in terms of maximum dollars that can be spent or types of service contracts that can be negotiated. An example interaction using this mechanism is described in Section 4.65.

3.3 Sentinel

Sentinels are mid-level rational software agents that, together with its managed host(s), correspond to the notion of autonomic elements. In autonomics parlance [9], the Sentinel is the Autonomic Manager and the host is the Managed Element. Each Sentinel is responsible for a single machine or group of machines that are similarly configured. For example, a Sentinel might be responsible for a single server, a router, a storage area network, a group of load-balanced web servers, or even a set of managed user workstations. Sentinels implement the policy they receive from the Sergeant and apply it to the configurations of the machine(s) they manage.

Sentinels also interface with the lowest-level agents: the swarming Sensor agents that gather information on potential problems found on the hosts. Sentinels provide the local geography to Sensors and provide mobility by negotiating with their destination. Sentinels also provide rewards and spawning capabilities to visiting Sensors. The Sentinels combine evidence from the Sensors with their own experience, shared knowledge from other Sentinels, guidance from Sergeants and Supervisors (interpreted by the Sergeant), and contextual host information to determine whether a problem exists and to devise potential solutions. Mechanisms similar to this are used in survivability architecture Willow [12].

Sentinels give feedback on the utility of Sensor findings in the form of “rewards” to the Sensors that visit their nodes. The analogy of foraging ants is used to describe the effect this feedback has on the system. By rewarding visiting Sensors, the Sentinel will attract more of them. A variety of visiting Sensors will provide more information on the potential problems experienced by the Sentinel and enable it to make more informed decisions about how to fix the problems.

3.4 Sensor

Sensors are lightweight, swarming, mobile software agents that roam and detect problems. They are modeled after behaviors of social insects and they employ a form of ant-colony algorithms and swarm intelligence [13]. The Sensors' logic is as simple as possible; their power is in their numbers and their diversity. Sensors wander across the geography superimposed on the enclave by randomly adjusting their current heading similar to the movement of real ants. Each Sensor uses a learning classifier [14] to match a particular set of conditions in the hosts they visit. There are two broad categories of Sensors: Markovian (memoryless) and differential. Markovian Sensors look for static conditions that may either define signatures of known problems or well-known anomalous conditions. Differential Sensors look for differences in conditions between hosts in recent memory and their current host. For example, there may be an unusual rate of network connections, a large number of open files, strange file names in system directories, or unusually high processor utilization.

Sensors communicate with each other stigmergically via trails of digital pheromone [5] messages. Decentralized, pheromone-based systems have been demonstrated to simply and effectively solve highly constrained problems where logic-based, optimizing approaches prove intractable [15]. Pheromone-based techniques have been shown to be robust and therefore appropriate for dynamic applications, such as network routing [15, 16].

Sergeants and Sentinels select successful Sensors (those that are consistently rewarded for useful findings) as templates for spawning new Sensors. This may be done by perturbing the parameters of a single Sensor’s classifier or by combining the classifiers of two or more Sensors.

4 Interaction Examples

CID is a mixed-initiative system that operates in a fundamentally different way from systems where initiative is solely human or solely automated. Initiative for actions, handling of interruptions, and responsibility for various activities are all shared. The purpose of CID’s design is to enable humans to function in the role they believe most appropriate at a given time. Thus, CID’s automation is decentralized and flexible, accommodating many types of interactions. In this section, we will highlight several of the

characteristic interactions designed into CID. These imply many more, but we have selected the following to highlight responsibilities within the system and to underscore the flexibility of human intervention they enable.

4.1 Supervisor-Sensor Interaction

The Supervisor can adjust global target parameters to make the Sensor swarm more or less responsive to intrusion evidence and control Sensor population. Two of these parameters are particularly important: activation and crowding tolerance. Sensors are programmed to try to achieve their target levels of these two quantities. The Supervisor may use these two parameters to regulate the size of the Sensor swarm and adjust its responsiveness to attackers. Activation level measures how successful a Sensor has been in collecting evidence useful to Sentinels. High activation yields larger Sensor populations and diverse classifiers. Crowding measures how often a Sensor encounters other Sensors while foraging. If a foraging Sensor senses crowding in excess of its target level, it is more likely to terminate itself reducing the population by removing Sensors that are no longer effective. Highly activated Sensors and Sensors following pheromone trails do not check their crowding metric because high concentrations of Sensors are desirable where problems are being detected. Thus the swarm adapts to new problems and maintains an appropriate size.

4.2 Sensor-Sentinel Interaction

Sentinels and Sensors cooperate to reduce the interruption false positives cause for human defenders. It is important to understand that even very low false positive rates can yield a tremendous amount of interruption that humans are not well equipped to handle [18]. Each false alarm that reaches a human implies an investigation may be necessary. Interactions between the Sensor swarm and the Sentinels weed out most of those false positives before they become alarms that interrupt the human. However, humans still need visibility into the system (Section 4.5), and the system may need human guidance to classify new types of input (Section 4.3).

Sentinels reward Sensors that find useful evidence by adding to their activation level. At activation levels below their target, Sensors remain in foraging mode, actively following pheromone trails and seeking evidence that will earn them a reward from Sentinels. When a Sensor's activation level exceeds the target, it stops foraging and begins dropping pheromone messages that point to the Sentinel where it was most greatly rewarded. As it drops pheromone, it loses activation until it enters foraging mode again. Consistently high activation levels indicate that a Sensor is successful and a Sentinel should select it for spawning.

As the Sentinel collects more and more information from various Sensors that visit (along with information from other sources, Section 4.3) it will develop either a diagnosis of the problem and a means to fix it or an understanding that this situation is actually an acceptable variation. The Sentinel will use this knowledge to decide how to feed future Sensors that visit. When the Sentinel has "enough information" (the reports from the Sensors fit a model) on the problem, it will make a report to its Sergeant. The report to the Sergeant will contain the classifiers used by the Sensors that helped diagnose the problem and any other evidence it has that may be useful in new classifiers.

False positives from the Sensors are not a problem; the Sentinel should feed them unless what the Sensor reports is *known* not to be a problem. This strategy will attract more Sensors until a clearer picture can be constructed. CID uses false positives to generate useful positive feedback and increase its problem-solving ability.

4.3 Sentinel-Supervisor Interactions

Sentinels use a semi-supervised, reinforcement-learning algorithm to classify evidence received from visiting Sensors. They dialogue with the Supervisor (mediated through the enclave Sergeant) to train themselves on problems that are too difficult to decide without human guidance. The untrained Sentinel classifies most kinds of evidence as unknown, asking for feedback from the human Supervisor on its decision. Asynchronous feedback from the human supervisor provides rewards and punishments to the Sentinel's reinforcement-learning algorithm. Sentinels may classify Sensor findings on a continuous scale (see Figure 3) from normal with high confidence (-1) to suspicious with high confidence (+1). Near zero, the Sentinel's classification is more uncertain. Low confidence metric values (close to zero) between the suspicious and normal certainty threshold values trigger the Sentinel to ask for clarifying feedback from the human. When the confidence metric is outside the certainty thresholds, Sentinels will share their classification information with neighboring Sentinels whose machines have similar architectures. In this way the system avoids having humans manually diagnose the same problem again and again.

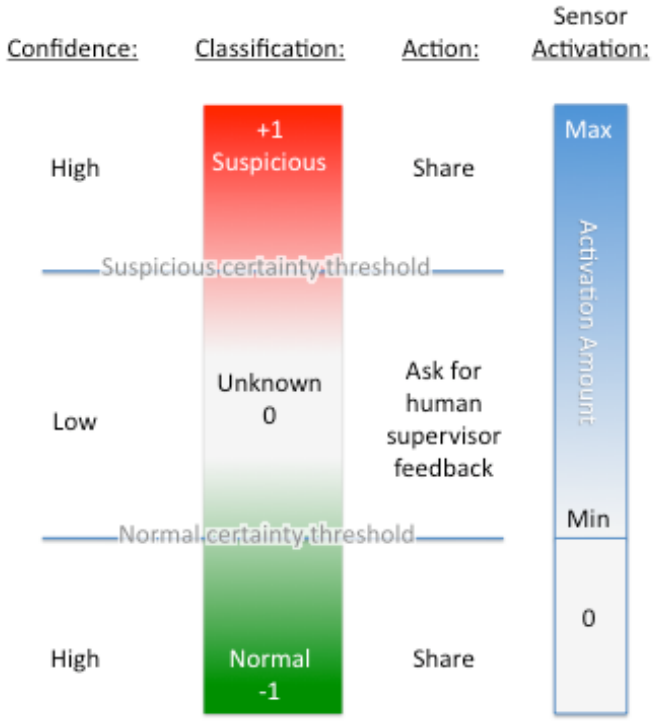


Figure 3: Sentinel classification of and actions based on Sensor data.

Trained Sentinels no longer require direct human feedback. Instead they use information gathered from the Sensors, from the Sergeant, and from other Sentinels to diagnose and repair the problem indicated. First, the Sentinel checks its local database of problems and solutions derived from its own experience and that of its neighbors for a near match. If no match exists, it requests matches from the enclave-level Sergeant agent. If no matching problems exist, the Sergeant will then alert the human Supervisor who can repair the problem manually or dismiss it as a non-problem. A special command shell or interface provided by the Sentinel is instrumented with learning classifiers and can learn how to repair similar problems by monitoring the human's repair activities. When a new known problem and solution are discovered, the Sentinel stores this information in its database and passes the solution up the hierarchy to the enclave-level agent, the Sergeant. The report will contain the classifiers used by the Sensors that helped diagnose the problem and any other evidence it has that may be useful in new classifiers.

4.4 Sergeant-Sentinel Interactions

Sergeants and Sentinels take charge of several areas to reduce human workload in managing large enclaves. Sergeants provide to the Sentinels the enclave security policy and a geographic representation of the enclave that defines the neighbors of each Sentinel. Sentinels provide reports to the Sergeant on problems that have been solved and on Sensor types that have been particularly effective. These interactions allow the human Supervisor to retain situational awareness and supervisory power but do not require the Supervisor to exert direct control to secure the system.

The two-dimensional geography the Sensors move around on is maintained by the Sergeant. The Sergeant builds its geography by periodically conducting a breadth-first search of the enclave and ordering the enclave members along a Peano-Hilbert space-filling curve. This curve preserves network distances as spatial distances on the grid and minimizes the number of hops a Sensor travels on its way to a neighboring Sentinel. As hosts join and leave the network, the Sergeant automatically adjusts and broadcasts the geography to keep it valid.

Sentinels report to their Sergeant about Sensors that have been especially useful for detecting problems on their hosts. The Sergeant uses this list of the most successful Sensors to create anticipatory classifiers based on combinations of highly successful classifiers. Sergeants may also share information from this list of Sensors with the Sergeants of allied enclaves.

4.5 Sergeant-Supervisor Interactions

The Supervisor interacts with the Sergeant through an information visualization and graphical user interface that gives the Supervisor situational awareness of state of the enclave. Utilizing the Netlogo agent simulation toolkit, we have simulated variations of this interface and plan to conduct user studies on which is most effective. Using slider bars, the user can adjust the parameters of the environment to influence the behavior of the agents involved. As mentioned earlier, the Supervisor uses this interface to adjust the activation and crowding tolerance target levels.

In the simulated interface (Figure 4), the Sergeant gives a color indication of the condition of the hosts. Each square in the visualization represents the status of a Sentinel that reports to the Sergeant. The color of the squares could indicate activity levels, security conditions, or any encoding the Supervisor prefers. In the simulation we use the red, green, and blue components of the color to represent file system, memory, and CPU activity levels. Systems performing similar tasks typically have similar colors. In the figure, workstations appear reddish, web servers are shades of blue, and file servers appear gray. A color change would indicate a change in behavior and function and could indicate a problem.

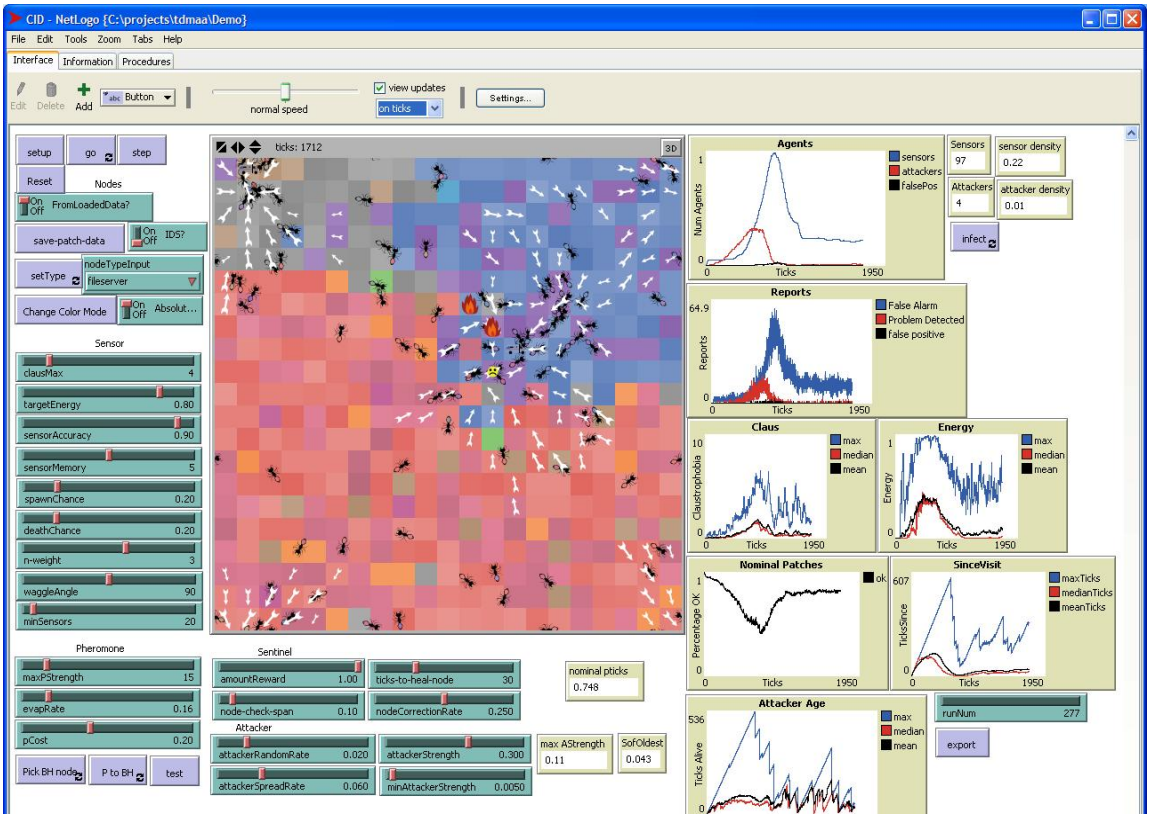


Figure 4: Prototype of Sergeant UI

Alternatively the coloring scheme could show the relative state of each host. A host running in a nominal state would be black. When a Sentinel reports a potential problem that needs human intervention, the Sergeant could color increase the red color in the square. The color could fade as the Sentinel resolved it, but an increasing intensity might indicate that the need for guidance is becoming more urgent. The Sergeant might then enact other means to interrupt the Supervisor and get attention to the troubled machine. It also might contact other neighboring enclaves to ask their Sergeants if similar problems are occurring there.

Another major Sergeant-Supervisor interaction is the establishment of policy for the enclave. We envision this to be a natural language and graphical interaction process where Sergeants and human Supervisors iteratively dialogue to arrive at policies that will enact the business and security objectives desired by the owners of the enclave systems. We anticipate a natural-language dialogue similar to that used by IBM's SPARCLE (Server Privacy ARrchitecture and CapabiLity Enablement) policy workbench [10] which translates natural language requirements into a standard machine readable form. Once the dialogue converges to an acceptable set of policies, the Sergeant will compile the Supervisor's guidance into executable policies that direct the Sentinels in their machine configurations.

Another important Supervisor-Sergeant interaction type involves the Supervisor delegating authority to the Sergeant up to specified authorization limits. Sergeants broker agreements between organizational units on behalf of the Supervisors, so they must have a separate digital identification from the Supervisor they report to. Because they negotiate on behalf of humans they may incur liability for their owning organization. Thus, there must be a mechanism to describe the degree of trust the Supervisor has placed in them. The trust mechanism may be implemented via authorization certificates such as the Simple Public Key Infrastructure (SPKI) certificate (RFC 2692, http://www.ietf.org/rfc/rfc2692.txt) which supports authorization and delegation up to specified limits. For instance, if a Sergeant wishes to negotiate a change in the Service Level Agreement (SLA) its unit has from an Internet Service Provider (ISP) the cost of the

change would be evaluated. The ISP could ask the Sergeant whether it had sufficient trust to negotiate the change autonomously. The Sergeant would present authorization credentials, signed by its Supervisor showing the dollar amount (or whatever unit) it is entrusted with to make decisions of this sort. If the authorization level was sufficient, the ISP would go ahead with the change. If the Sergeant lacked sufficient authorization, the ISP could require the human Supervisor's signature on the change. The Supervisor would probably start out trusting the Sergeant very little, but as the Sergeant gained experience and demonstrated reliability, the Supervisor might increase its delegated authorization limits.

4.6 Sergeant-Sergeant Interactions

Sergeants are the only software agents that communicate with entities external to the enclave such as outside resources and other Sergeants. Outside resources such as virus definitions and attack profiles would assist the Sergeant in formulating guidance for the Sentinels. An updated attack profile would be sent to all Sentinels in the system so they can begin watching for them based on Sensor reports. Likewise, if a new attack is identified and resolved on the system, the Sergeant would pass this information along to allied Sergeants in neighboring enclaves (those it trusts based on the mechanisms in 4.5). The information passed might be a simple description or it might contain parameters and classifiers actually used by the Sensors that helped the system identify the novel attack (or effectively deal with known attacks). Sergeants in charge of other enclaves could then instantiate these Sensors in their own enclaves. A benefit of this approach is that the Sensor definition can be shared without sharing the data from any machine within the enclave and avoiding data exfiltration or violation of proprietary information.

4.7 Sensor Agent Management

Decentralized, pheromone-based systems have been demonstrated to simply and effectively solve highly constrained problems where logic-based, optimizing approaches prove intractable. Figure 4 shows the simulation model of a pheromone based system we built to investigate tradeoffs among pheromone implementations. Central to any system that relies on swarming sensors is the ability to communicate information and influence action. The CID framework uses a pheromone-based system for inter-Sensor communications that has been demonstrated to be simple and efficient [15]. This decentralized communication approach allows Sensors to not only achieve the local goal of finding food, but also collectively achieve the global objective of discovering inflected computers in a dynamic environment.

Pheromone-based techniques have been shown to be robust and therefore appropriate for dynamic applications, such as network routing [15][16]. This is an important feature for CID since the number of computers and their security status (*e.g.* infected or clean) will change over time. As previously discussed, a Sensor that discovers pheromone has a probability of following it that is related to the pheromone strength. The rate of pheromone decay and how pheromone is potentially overwritten will impact the system adaptiveness, as discussed in [15][16].

5 Related Work

The mixed-initiative approach is not common in cyber security applications. CID blends the initiative of humans and a hierarchy of agents to avoid the extremes of either marginalizing the human in favor of black-box machine intelligence or including the human in the wrong loop, too close to the problem so that the system speed is bounded by human reaction time.

In earlier research [20] interviews with operators of computer systems we interviewed agreed that a "five nines" (99.999%) reduction of information was necessary for them to retain situational awareness. System administrators and security officers we interviewed indicated that their preference was to see changes in the visualization only when something that required human attention occurred. Feedback from other clients revealed that operators find even a hundred alerts per day too taxing. Clearly, operational staff need to reduce the amount of workload required by the available data, and teaming with autonomous agents is an excellent potential solution.

The hierarchical arrangement of humans and various types of agents has been proposed in a variety of forms. Ibrahim [17] proposes a network management system that utilizes a hierarchy of mobile agents to manage a distributed system while reducing bandwidth consumption. Parunak [6] proposes a heterogeneous hierarchy to solve highly constrained military movement problems. However, the human mostly serves as an observer in these systems and has very limited interaction with the agents.

The network management system proposed by Ray [19] proposes a similar arrangement of agents as well as keeping the human as a high level policy maker. Our system is further decentralized with no concept of a "home base" for our mobile (and non-cognitive) Sensors.

6 Conclusions and Future Research

Thus far we have focused our research mainly on the Sentinel and Sensor behavior in our simulated environment thoroughly mapping out the behavior of the Sensors and their interactions with the Sentinels. We have created a demonstration system of the Sensor-Sentinel levels in JADE and have created vertical prototypes of interactions at the Supervisor, Sergeant, and Sentinel levels. However, we have several areas of future research we intend to pursue such as:

- Defining the dialogue between Supervisor and Sergeant that translates human guidance into unambiguous policy for the Sentinels
- Experimenting with our practical Sensor-Sentinel demonstration on a computer cluster
- Conducting usability studies on the prototype Sergeant user interface
- Learning to generalize knowledge gained from solutions to problems that share similarities
- Testing the framework against real-world attacks on the systems being monitored

The CID framework keeps the human in the right loop via a hierarchy of software agents that implements high-level policy and handles low-level events. The framework resolves issues in a decentralized manner at the lowest possible level to minimize alerts that interrupt the human user. This builds user trust in the system and improves system efficiency. The efficiency realized by CID's mixed-initiative approach is highly scalable and difficult to disrupt. We believe that by sharing system control with a hierarchy of agents and utilizing the truly adaptive capabilities of swarm intelligence we can greatly enhance the security of our computational infrastructures and the societies that depend on them.

References

[1] Allen, J.E., Guinn, C.I., Horvtz, E.: Mixed-Initiative Interaction. IEEE Intelligent Systems. 14, 14—23 (1999)

[2] Frincke, D., Wespi, A., Zamboni, D.: From Intrusion Detection to Self-Protection. Computer Networks. 51, 1233--1238 (2007)

[3] Dahiyat, E.A.R.: Intelligent Agents and Intentionality: Should We Begin to Think Outside the Box? Comput. Law Secur. Rep. 22, 472--480 (2006)

[4] Scher, M.B.: On Doing "Being Reasonable." ;login: 31, 40--47 (2006)

[5] Brueckner, S.: Return from the Ant: Synthetic Ecosystems for Manufacturing Control. PhD Dissertation. Department of Computer Science, Humboldt University, Berlin, (2000)

[6] Parunak, H.V.D., Nielsen, P., Brueckner, S., Alonso, R.: Hybrid Multi-Agent Systems: Integrating Swarming and BDI Agents. In: Brueckner S.A.., Hassas, S., Jelasity, M, Yamins, D. (eds.) ESOA 2006. LNAI, vol. 4335, pp 1--14. Springer, Heidelberg (2007)

[7] Smieja F.: The Pandemonium System of Reflective Agents. IEEE Transactions on Neural Networks. 7, 97--106 (1996)

[8] Selfridge, O.G.: Pandemonium: a Paradigm for Learning. In: Anderson, J.A.D., Rosenfeld, E.: Neurocomputing: Foundations of Research, pp 115--122. MIT Press, Cambridge, MA, USA (1988)

[9] Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. Computer 36, 41--50 (2003)

[10] Karat, J., Karat, C.-M., Brodie, C., Feng, J.J.: Privacy in Information Technology: Designing to Enable Privacy Policy Management in Organizations. Int. J. Human.-Computational. Studies. 63, 153--174 (2005)
[11] Rao, A.S., Georgeff, M.P.: BDI Agents: From Theory to Practice. In: First International Conference on Multi-Agent Systems (ICMAS-95). pp. 312--319. AAAI Press, Menlo Park, CA, USA (1995)
[12] Knight, J., Heimbigner, D., Wolf, E.L., Carzaniga, A., Hill, J., Devanbu, P., Gertz, M.: The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications. In: Distributed Applications: Intrusion Tolerance Workshop, Dependable Systems and Networks (DSN 2002). Washington DC (2002)
[13] Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans. Evolutionary Computation. 1, 53--66 (1997)
[14] Holland, J.H., Booker, L.B., Colombetti, M., Dorigo, M., Goldberg, D.E., Forrest, S., Riolo, R.L., Smith, R.E., Lanzi, P.L. Stolzmann, W., Wilson, S.W.: What Is a Learning Classifier System? In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) Learning Classifier Systems '99. LNCS, vol. 1813 pp. 3--32 Springer, Heidelberg (2000)
[15] Di Caro, D., Dorigo, M.: AntNet: Distributed Stigmergetic Control for Communications Networks. J. Artifificial Intelligence Research. 9, 317--365 (1998)
[16] Bonabeau, E., Henaux, F., Guérin, S., Snyers, D., Kuntz, P., Theraulaz, G.: Routing in Telecommunications Networks with "Smart" Ant-Like Agents. Working Papers 98-01-003, Santa Fe Institute. (1998)
[17] Ibrahim, M.A.M.: Distributed Network Management with Secured Mobile Agent Support. In: Proceedings – 2006 International Conference on Hybrid Information Technology, ICHIT '06, vol. 1, pp. 244--251. IEEE, Piscataway, New Jersey (2006)
[18] Axelsson, S. The base-rate fallacy and the difficulty of intrusion detection. ACM Trans. Information and System Security, 3(3):186–205, 2000.
[19] Ray, P., Parameswaran, N., Lewis, L., Jakobson, G.: Distributed Autonomic Management: An Approach and Experiment Towards Managing Service-Centric Networks. In: 5th International Conference on Service Systems and Service Management, ICSSSM'08, pp. 1--6. IEEE, Piscataway, New Jersey (2008)
[20] Fink, G.A., et al. Bridging the Host-Network Divide: Survey, Taxonomy, and Solution. in In Proceedings of the 20th Large Installation System Administration Conference (LISA '06). 2006. Washington, DC: USENIX.

Agent Support for Human Team Collaboration in Uncertain Environments*

Daniele Masato, Timothy J. Norman, and Wamberto W. Vasconcelos

Department of Computing Science
University of Aberdeen, AB24 3UE, Scotland, UK
{d.masato,t.j.norman,w.w.vasconcelos}@abdn.ac.uk

Abstract. Working collaboratively in a team and formulating a plan of action to achieve a particular goal is often a complex task for humans, especially when the decision-making process is performed in time-stressed situations, in which response teams are assembled in an ad-hoc fashion and operate in dynamic and uncertain environments. We are conducting research towards the development of software agents that will track the activities of human teams and monitor the plan execution in order to offer advice that would enable the humans to avoid problems, resolve them or recognise unexpected options, and in general maintain the synchronization among the different plan components.

Key words: Agent Support, Human Team, Plan Synchronisation, Uncertain Environments

1 Introduction

Working together in a team and formulating a plan of action to achieve a particular goal is often a complex task for humans. This is mainly because making decisions, taking responsibilities and performing joint actions all involve a number of interdependent activities that need to be coordinated together. These issues are emphasised when the decision-making process is performed in time-stressed situations, in which response teams are assembled in an ad-hoc fashion and might be asked to perform non-standard tasks in dynamic and uncertain environments. In such a context, humans may be overwhelmed by the amount of information they need to process to perform decision-making and coordination of tasks; thus, humans may fail because they do not recognise that initial

* This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation hereon.

assumptions are no longer valid and the actual course of action needs to be revised.

It is accepted that, at the moment, humans are much better than any intelligent software in analogical, spatial and heuristic reasoning, yet they are on the other hand limited in their cognitive ability to process large amounts of information [5]. Thus, when under stress, humans would make better and faster decisions if the *right information* was delivered to the *relevant recipient* at the *right time*, rather than having to process and share large amounts of information on their own. Software agents can support the information-driven decision-making process by constructing and maintaining a model of the plan and the context the team is operating within [8]. The model can be exploited to understand humans' intent and activities, recognise problems in advance and potentially offer advice that would enable the team to avoid problems, resolve them or recognize unexpected options, in this way relieving the team's cognitive burden.

The remainder of this paper is organised as follows: in Section 2 we outline our ideas to design software agents which can provide the type of support mentioned above. In Sections 3 and 4 we describe what kind of knowledge software agents need to acquire in order to monitor the plan execution and how that knowledge can be represented into the agents. In Sections 5 and 6 we present our approach to the agent design, starting with efficient ways to recognise team activities, and map them to the actual team's intent, of course considering plan constraints and the uncertainty of the environment. We finally conclude with some related work and future directions in Sections 7 and 8.

2 Research focus

Recognising human teams' intent and activities in a realistic simulation is the first step towards the design of software agents that are aware of the team situation and can monitor the plan execution, providing the kind of up-to-date, timely advice required to enhance team collaboration and human-agent interactions. Figure 1 presents our approach to tackle this problem. We assume that a number of human teams are collaborating together on a plan to achieve an overall goal (left-hand side of the figure). The main goal is divided into multiple subgoals, each tipically assigned to a specific team. An assigned subgoal $Subgoal_i$ is in turn achieved by executing in a number of hierarchical tasks $Task_i$, each of those can be recursively subdivided in simpler subtasks $STask_i$, in a *Hierarchical Task Network (HTN)* [9] representation. The bottom level of a task decomposition consists of a set of group behaviours B_i (i.e. manoeuvres in formation, see Section 3.2) the team is assumed to execute as part of the expected course of action (*Expected COA* in Figure 1).

The achievement of the (sub)goals can be affected by dependencies among the tasks (see Section 4.1) that have to be satisfied during the execution. For example, in Figure 1, $Team_1$ and $Team_2$ are collaborating in parallel, however their subtasks ($STask_i$) need to follow a sequence (arrows between boxes). In a dynamic and uncertain environment, the synchronisation among the different

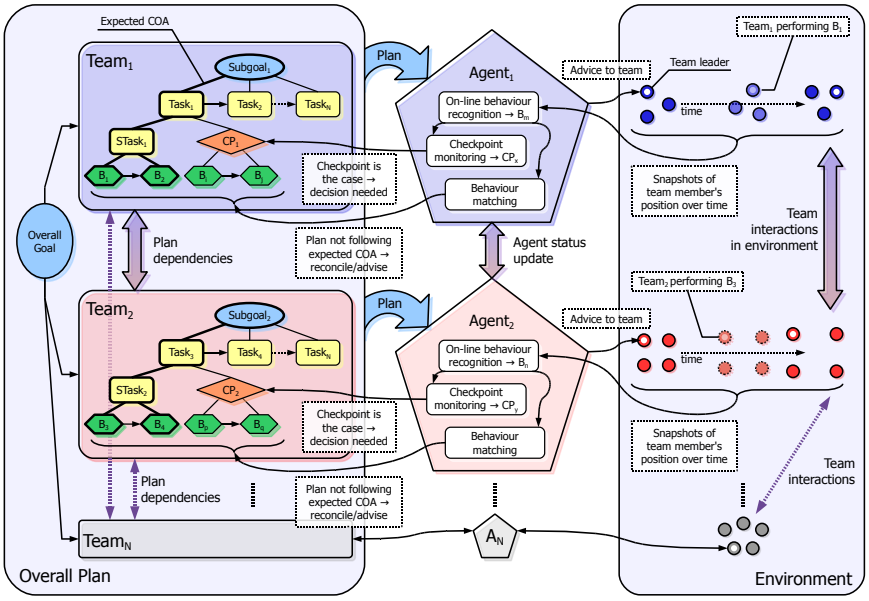


Fig. 1. System overview

plan components might be difficult to maintain, or changes in the course of action may become necessary, hence planners usually employ checkpoints CP_i (see Figure 1) to monitor the execution and identify problematic situations (see Section 3.3). Checkpoints can also be used to pinpoint conditions that could positively impact the plan, as shortcuts and new opportunities. Because the teams interacting within the environment (right-hand side on the figure) might not follow the expected course of action, it would therefore be appropriate to reconcile their behaviours with a component of the plan, if possible. For example, because of terrain constraints, $Team_1$ and $Team_2$ might have to swap their roles during the execution of the plan without jeopardising the plan outcome as long as both $Subgoal_1$ and $Subgoal_2$ are achieved. This requires that both teams are aware of the change to avoid losing synchronisation, but this might be prevented because, say, there are visual obstacles or radio communication problems.

We aim to maintain and improve the collaboration between teams by attaching to each of them a software agent performing those monitoring processes that would usually be carried out by humans, but may result problematic in the circumstances mentioned above. By analysing on-line spatio-temporal traces of the teammates' positions in the environment, the software agent can recognise group behaviours and match them with the expected course of action, checking

at the same time whether checkpoints are being satisfied[1]. When the enactment is not following the expected track, we envisage software agents to try and reconcile the evidence with the plan, possibly exchanging status updates among them, or eventually warning the team leader that his group is off-sync and a decision needs to be taken. The same intervention mechanism could be used to offer advice about new opportunities, e.g. estimating whether a goal can be achieved faster because a set of preconditions has become the case.

3 Information requirements for software agents

Today, one of the challenges of assisting humans in their activities does not lie in the capabilities or power of the computer or device to be used, but rather in having the humans communicating their intents to a software agent and being confident enough to trust the agent's reply [15]. The risk is indeed that an inappropriate agent's intervention could be dismissed as irrelevant or annoying, defeating the efforts of supporting humans. Nonetheless, in order to understand what is happening in the environment, software agents need to gather information about the plan being pursued and its development. In the following discussion, we examine how much knowledge we can collect both during the plan design and execution.

3.1 Plan design phase

When teams operate in an uncertain environment, it might be difficult, if not impossible, to formulate a plan that foresees every possible contingency (e.g. in a military domain the classic assumption *"no plan survives contact with the enemy intact"* is often cited). This is worsened by the lack of time that can arise even during the design phase and because of non-standard tasks that need to be carried out. As a consequence, the plan structure may not cover in detail some of the tasks and several contingencies may be missed, under the hypothesis that an experienced team will be able to cope with these issues as the enactment proceeds. This usually works when human teams are well trained and have been working together before, but may fail for ad-hoc instances, which are formed rapidly and without much time for co-training.

Nevertheless, planners do lay out an expected course of action (see *Expected COA* in Figure 1), which can be represented as the agent initial knowledge and exploited for the monitoring purposes. Other opportunities to gather information arise, even in time-critical situations, when humans are given the chance of incrementally define/modify their plans [1] during the enactment phase, and if agent asked for more input only when such input would be relevant to the decision-making process [15], so we will investigate techniques which will enable us to incrementally modify a designed plan.

[1] For simplicity, we will initially assume the agents know the whole plan with all its dependencies

3.2 Plan execution phase

In circumstances in which time is limited and the team is actively involved in the dynamics of the plan execution, speech is probably the most effective and immediate means of communication. Unfortunately, performing speech-to-text analysis of communications in a hasty execution environment, with overlapping voices and noise, is still a big challenge, so we shall not employ any text-to-speech technology for our agents.

On the other hand, there are basic group behaviours that human teams employ to react to the surrounding environment, both in planned and unexpected circumstances (e.g. formations in group sports or military manoeuvres, see Figure 1, right-hand side). These behaviours have distinctive spatio-temporal structures, in terms of the relative position of the team members to each other and to external landmarks, and in terms of the temporal sequence within which the spatial configuration is maintained. These patterns can be exploited to design algorithms that recognize such behaviours [13] (see Section 5). Basic team behaviours also represent what the team is assumed to have learned through previous training or agreements, and can be considered the building blocks for a more complex plan, composed in a hierarchical fashion.

3.3 Plan execution monitoring

Once the plan has been designed and its enactment starts, monitoring and assessing this enables humans to react to changes, unexpected failures or new opportunities. In general, checkpoints are put in place to focus the monitoring process and detect whether critical decisions need to be made. They are established in the plan design process (but we could foresee them being added during the execution phase as well), to help performing timely and relevant decisions. A *timely* decision allows overcoming problems of the environment physics, giving enough time to the team to position itself in relation to the environmental constraints (e.g. terrain) and other teams to achieve the desired task. A *relevant* decision allows gaining and maintaining the ability to achive the plan goals. Checkpoints are usually employed to decide whether to employ time sensitive/critical assets in the context of the current plan, deviate from the original plan (alternative courses of action or branches) or initiate the transition to the next plan task.

A way to define a checkpoint consists in a *set of conditions* that, when they hold, triggers the decision-making process. The preconditions can be specified as conjunctions of observations, based on the information coming from the environment or from the plan execution status. A checkpoint also defines a *deadline* by which the information required to evaluate the conditions needs to be obtained, in order to perform a timely decision. After that, the decision would no longer be relevant. Note that a checkpoint may not define the precise details of the course of action to be taken (they could be specified when that particular situation occurs), nonetheless it identifies a point of contact between what the team leader would like to know about the plan while it is being executed and what a software agent can provide to try and satisfy these requirements.

Example. In military planning, checkpoints go under the name of *Decision Points*. The Commander sets a number of Decision Points during the planning process and for each he specifies the *Commander's Critical Information Requirements (CCIR)*, the requirements which *"allow the commander to track potential decisions, recognize that the time is right and that the decision is relevant to overall success"*[2]. A *Latest Time Information Is Of Value (LTIOV)*, referred above as the deadline, is also attached to a Decision Point.

4 A minimal model of context

As introduced in Section 1, software agents can support the decision-making process by constructing and maintaining a model of the plan and the context the team is operating within. We will now describe a minimal model of context that would allow us to design such agents.

4.1 Plan representation

n our research we will initially represent a plan in a hierarchical way using the concepts and control construct provided by the OWL-S[3] ontology. There is a straightforward link between *Hierarchical Task Network*-style plans [9] and the OWL-S process model, such that most of OWL-S concepts can be directly mapped to HTN constructs [16, 12] which an HTN planner can process. The plan hierarchy will include both the recursive plan decomposition into atomic tasks and the temporal relationships between the decomposed tasks (ordering, synchronization, etc.) made possible by the following OWL-S control structures[4]:

- **Sequence**: defines a list of tasks that need to be executed in order.
- **Any-Order**: defines a list of operations that can be done in any order.
- **Split**: defines a set of tasks that should be executed concurrently.
- **Split+Join**: defines a set of concurrent tasks that will terminate only when all the components have completed.

Each atomic task generated by the hierarchical plan decomposition process (B_i in Figure 1), should be defined in terms of:

- **What**: the group behaviour expected from the team (the atomic task itself).
- **Who**: the team who is scheduled to perform the *What*. This parameter may be optional, as sometimes the decision about who will do what is made during the actual execution.
- **Where**: the location in which the *What* is to be performed.

[2] See `http://www.armchairgeneral.com/tactics-101-014-decision-making-and-the-power-of-commanders-critical-information-requirements.htm`

[3] See `http://www.w3.org/Submission/OWL-S`

[4] OWL-S also specifies the **Iterate**, **Repeat-While** and **Repeat-Until** control constructs but at the moment we will only consider non-iterative structures.

- **When**: an expected time (interval or point in time) by which the *What* should be completed[5]. This parameter is not required, but it is probably the most important in time-stressed situations.
- **Which**: the resource used to performed the *What* (e.g. a particular vehicle or asset the team will employ). This may be optional as well.
- **Why**: the reason why the *What* needs to be carried out. The justification for a task definitely helps in providing better advice to the team during the execution, especially when the course of action changes.

It is not necessary to specify all of the above parameters at the lowest level of the hierarchy, for each atomic task. Instead, some of them may be specified at the topmost or intermediate levels within the hierarchy, and then inherited by the levels underneath (e.g. *Who* could be specified for a complex task – say, a Sequence – so that all its subtasks would be performed by the same team, if no more specific information are given for the subtasks).

OWL-S also defines the concept of **Choice**[6], a set of tasks from which one should be chosen and executed. OWL-S Choices can be seen as checkpoints in our plan representation (see Section 3.3), where different choices may represent different courses of actions. The preconditions of a choice can be listed as:

- **What** behaviour is being performed in order to consider the checkpoint.
- **Who**, **Where** and **Which** as defined above.
- **When**, that is an expected time (interval or point in time) by which the *What* is expected to happen. This parameter can be optional and is not necessarily related to the deadline, however it must not conflict with it[7].

4.2 Information collection during execution

In order to infer the team's intent, a software agent needs also to collect information while the plan is being carried out. Because in some environments it might be difficult to employ extensive instrumentation (e.g. cameras, sensors, radios) to monitor the plan execution, we will initially rely only on spatio-temporal observations of the team members' positions collected at regular intervals over time, with an option to integrate more data sources in the future. Retrieving the position of a moving person or vehicle in a simulated environment is quite easy, but it should be feasible in real-life exercises as well, by simply using a GPS device (which nowadays can be found in many high-end mobile phones).

Similarly to the above discussion, the execution information can be seen as:

- **Where**: the position of a team, can be calculated as the centroid of the team members' positions.

[5] This should of course be consistent with the plan structure as specified by the combination of OWL-S control structures. For example, the second task in a Sequence cannot have an expected time set before the first task's one.

[6] We will consider the OWL-S **If-Then-Else** construct as a special case of **Choice**.

[7] For example, if the *When* is defined as an interval $[a, b]$, then $b < deadline$ must hold, whereas if it is an expected time t then $t < deadline$ must hold.

– **When**: the instant in time in which the position is sampled.
– **What** and **Who**: because team behaviours show distinctive spatio-temporal patterns, the *When* and *Where* will be exploited to recognise what task the team is performing (see also Section 5).
– **Which**: the asset in use by the team. This could be tricky to identify, but we can again assume that it can be fitted with an identification tag.

In conclusion, using a minimal context model consisting of **What**, **Who**, **Where**, **When** and **Which** (5W) it should be possible to characterise the plan tasks, checkpoints and observations from the environment. This model, however, does not capture the whole context in which the plan is being enacted (hence the name "minimal"). For example it does not account for any communication among the team members or how the weather can affect the execution, but it should be simple enough to keep the computational cost low.

5 Proposed approach

The problem of monitoring a plan execution and matching recognised team behaviours to steps in the plan can be divided into three steps (see the boxes inside the Agent entities in Figure 1):

1. Acquiring regular "5W snapshots" of the environment using the 5W model.
2. Classify the current snapshot (or the sequence so far) to retrieve the team behaviour over time.
3. Using the results from step 2 as evidence, match it to the current plan execution status and identifying satisfied checkpoints.

Each of these steps is detailed in the following discussion.

5.1 Acquiring environment snapshots

We have chosen Battlefield 2[8] as our environment, a first-person shooter game with strategy elements in which players act in a virtual battle space. Players are organized in two teams and can play different military roles, use different types of weapons and particular classes of vehicles, depending on the selected role. We instrumented the game to capture many types of events, such as participants joining the game, interacting with vehicles and other players or entering in pre-defined areas identified by sensors [6]. The instrumentation allows us to collect *Who* is participating in the game, the participants' positions (*Where*) over time (*When*, the sampling interval can be changed), which assets (vehicles) they are using (*Which*), so that only the *What* needs to be inferred from the collected data.

[8] See `http://www.ea.com/official/battlefield/battlefield2/us`

5.2 Behaviour Classification

An interesting approach in recognising team behaviours (*What*) is presented in [13]: that work exploits the team members' positions to perform group behaviour recognition using traces collected from simulations. Specifically, an efficient algorithm called *Simultaneous Team Assignment and Behaviour Recognition (STABR)* is introduced. This algorithm analyses the traces in three stages:

- First the static positions of all the subjects in the environment are used to recognise possible formations, identifying in such a way also hypothetical teams.
- Each hypothetical team formation is then retained only it shows sufficient temporal support, that is, when such formation is following a known movement pattern over time (this also enables to identify the behaviour).
- Finally, the surviving hypotheses are used to partition the subjects into the actual teams over the time sequence, according to a cost function that gives higher scores to solutions with fewer changes in composition and behaviour of the teams.

STABR has proven to outperform agglomerative clustering, which is usually employed to group subjects into teams, even with noisy observations. One of the limitations in that work, however, is that the algorithm has been evaluated only with fully available (i.e. offline) traces, although real traces, rather than on-line observations that would be continuously gathered in our scenario. We are investigating means to adapt and extend *STABR* in an on-line setting to continuously classify the team's actions.

5.3 Plan status monitoring

In dynamic, time-stressed environments, it is inevitable to come across uncertainty, hence this aspect needs to be factored into the design of monitoring software agents which will be operating in those circumstances. Two main types of uncertainty are identified in [10]:

- *Aleatory*, that is uncertainty generated by a system behaving in random ways (also called objective uncertainty).
- *Epistemic*, that is uncertainty generated by the lack of knowledge about the system under analysis (also called subjective uncertainty).

The usual approach to handle aleatory uncertainty is the probabilistic (i.e. Bayesian) model, whereas for epistemic uncertainty the *The Dempster-Shafer Theory (DST)* [11] is often considered. The probabilistic model has been applied to epistemic uncertainty as well, however it has the following disadvantages:

- the a-priori probability of every event (e.g. performing a specific behaviour) is required. When such information is not available, a uniform probability is usually assigned to all events. This means deliberately assigning a probability to events whose characteristics/models are actually not known.

– The sum of all probabilities must add up to 1 and, as a consequence, knowing the probability $P(A)$ of an event A occurring entails knowing the probability $P(\neg A)$ of that event not occurring. This leaves out the concept of ignorance, that is, the fact that one may not know $P(\neg A)$.

The DST, instead, seems more suitable to handle epistemic uncertainty, particularly in situations where:

– It is not straightforward to assign probabilities to single events due to ambiguous, conflicting or lack of information (e.g. sensor readings).
– It is appropriate to consider subjective degrees of belief on sets of events, rather than on objective probabilities about single events.
– Knowing the likelihood of a set of events to happen does not necessarily entails knowing the likelihood of that not happening.

In our case, we can consider $S = (Ev, Pl, Tm)$ to be the system composed by the uncertain environment Ev, the plan Pl and a human team Tm executing Pl in Ev, where Ev accounts for other teams operating in the same area. Although Ev also consists of random components (e.g. the weather), they are very difficult to model in terms of probability. It is even harder to model how Tm would react to changes in Ev in terms of conditional probabilities. Moreover, Pl is not a random component of the system as it has been laid out beforehand, albeit in an abstract form, and teammates will not behave randomly, rather they will act deliberately both when collaborating with others and facing unexpected events. Therefore, we are of the opinion that it is appropriate to employ the DST in our domain. This translates as the fact that, when performing behaviour recognition, we do not need the probabilities for each possible behaviour, rather we can assign a degree of belief only to a subset of likely behaviours.

6 Behaviour matching using the DST

To perform the matching between behaviours detected in the environment and the expected course of action, we will follow closely the method introduced in [2], although we will use a different rule of combination. Given a finite set of hypotheses Θ whose power set is 2^Θ, the DST defines a *basic probability assignment (bpa)* as a function $m : 2^\Theta \mapsto [0, 1]$ where:

$$m(\emptyset) = 0 \quad \text{and} \quad \sum_{X \subseteq \Theta} m(X) = 1 \tag{1}$$

Hypothesis sets X for which $m(X) > 0$ holds are called *focal elements* of the *bpa*; $m(X)$ represents the amount of belief committed in exactly the hypothesis subset X, and not to more specific subsets of X. Thus, the total belief committed to X and its subsets is given by:

$$Bel(X) = \sum_{Y \subseteq X} m(Y) \quad \text{where} \quad X \subseteq \Theta \tag{2}$$

In our approach, each hypothesis in Θ will represent one of the atomic behaviours generated by the hierarchical decomposition, whereas a set of hypotheses will represent a branch of the plan (e.g. the expected course of action). For example, according to Figure 1, $\Theta = \{B_1, B_2, ..., B_i, B_j\}$ and $STask_1 = \{B_1, B_2\}$. Given a plan branch X, $Bel(X)$ will represent the amount of belief that the team is actually performing one of the atomic tasks in X. For example, according to Figure 1, $Bel(Task_1) = Bel(\{B_1, B_2, ..., B_i, B_j\})$ will evaluate whether the team is performing one of the expected behaviours in the actual course of action.

The DST also states how multiple, independent sources of evidence (degrees of belief on sets of events) should be combined to obtain an aggregate event likelihood, based on Dempster's rule of combination [11]. Given two *bpa*s m_1 and m_2, their combination is calculated as:

$$m_{12}(X) = \frac{\sum_{Y \cap Z = X} m_1(Y) \cdot m_2(Z)}{1 - c} \quad \text{where} \quad c = \sum_{Y \cap Z = \emptyset} m_1(Y) \cdot m_2(Z) \quad (3)$$

In Formula 3, c quantifies the *conflict* between two evidential sets and it is used to normalise the resulting *bpa* m_{12} so that both constraints in Formula 1 hold. This normalisation, however, has been disputed by several people ([10] reviews their work) because it completely removes the conflict and can yield counterintuitive results. Several alternative rules have been proposed, but there seems still to be no general agreement on any particular rule; rather, there are different lines of thought about how to combine new evidence according to the application domain. Given a *bpa* m_C describing the current assessment of the plan execution status and an environment snapshot (evidence E), we have identified three cases that can arise when the agent tries and detect a behaviour[9]. With reference to $Team_1$ in Figure 1, $Agent_1$ may:

1. Recognise a behaviour B_i which can be mapped to $Team_1$'s subplan (e.g. B_1, B_2). In this case the agent will produce a *bpa* m_E, based on the evidence E, where $m_E(B_i) = k$, $m_E(\Theta) = 1 - k$.
2. Recognise a behaviour B_p which cannot be mapped to $Team_1$'s subplan (e.g. B_3, B_4, assuming $Team_1$ and $Team_2$ have swapped their assignments). The agent will produce a *bpa* m_E where $m_E(B_p) = k$, $m_E(\Theta) = 1 - k$.
3. Not recognise any behaviour at all. In this case the agent will produce a *bpa* m_E where $m_E(\emptyset) = k$, $m(\Theta) = 1 - k$.

After the recognition, m_C will be combined with m_E to obtain a new assessment m_{CE}. There are a few considerations to make about the above situations:

– The parameter k, $0 \leq k \leq 1$, accounts for the accuracy of the *STABR* algorithm. *STABR* needs to perform several iterations to recognise a behaviour

[9] The issue of how to initialise m_C when no evidence has been collected yet needs to be considered as well. Although the approach in [2] suggests using the vacuous hypothesis $m_C(\Theta) = 1$ when no evidence is available, we are investigating other initialisation methods. These will assign initial values to $m_C(B_i)$ according to how atomic behaviours are structured and linked in the expected course of action.

with a given accuracy (e.g. 99%, $k = 0.99$), but a lower number might be required in order to increase the recognition speed in a real-time simulation.
- In all three cases, assignign a degree of belief of $1-k$ to the set Θ accounts for our ignorance about what behavior is being (or not) performed, according to the selected accuracy for *STABR*. Moreover, associating degrees of belief either to a singleton set or the set Θ of all hypotheses, reduces the computational complexity of combination rules from exponential time to polynomial time in the number of focal elements in the *bpa*s to be combined (see the *Singleton DST* introduced in [3]).
- In case 3, we want the resulting $Bel_{CE}(X)$ function calculated over m_{CE} to show a value below 1 for every $X \subseteq \Theta$, including the case $X = \Theta$. This would be indeed a strong signal that something is going wrong during the execution. To achieve this, however, we need to relax the definition of *bpa* (see Formula 1) by allowing $m(\emptyset) \geq 0$, such that $Bel(\Theta)$ will not account for some of the total belief mass, as it has be assigned to the empty set.

Among the several combination rules for the DST reported in [10], none seems to satisfy all the above requirements. For example, *Yager's rule* requires Formula 1 to hold and assigns the amount of conflict c to the set Θ (effectively transforming conflict in ignorance), so we cannot deal with situations covered by case 3. On the other hand *Smets' rule* always assigns conflict to the empty set, as a consequence if $Team_1$ starts performing B_1 but then switches to B_2 for any reason, the belief mass intially assigned to B_1 will be transferred to the empty set, resulting in $Bel_{CE}(\Theta)$ evaluating to less than 1 even if in fact $Team_1$ is still operating within the expected course of action. For these reason, we will employ the *Combination by Compromise (CBC) rule* [17]. Given two *bpa*s m_1 and m_2, this rule, based on Equation 3 with c set to 0, assigns a part of $m_1(Y) \cdot m_2(Z)$ to $C = Y \cap Z$, and distributes the remaining part to $Y_Z = Y \cap \neg Z$ and $Z_Y = \neg Y \cap Z$, depending on the values of $m_1(Y)$ and $m_2(Z)$, respectively. The combined *bpa* m_{12} is calculated as:

$$m_{12}(X) = r_1(X) + r_2(X) + r_3(X) \tag{4}$$

where

$$\begin{aligned}
r_1(X = C) &= \sum_{C=Y\cap Z} \left[n_1(C) \cdot n_2(C) + \frac{n_1(Y_Z) \cdot n_2(C)^2}{n_1(Y_Z) + n_2(C)} + \frac{n_1(C)^2 \cdot n_2(Z_Y)}{n_1(C) + n_2(Z_Y)} \right] \\
r_2(X = Y_Z) &= \sum_{Y_Z=Y\cap\neg Z} \left[\frac{n_1(Y_Z)^2 \cdot n_2(C)}{n_1(Y_Z) + n_2(C)} + \frac{n_1(Y_Z)^2 \cdot n_2(Z_Y)}{n_1(Y_Z) + n_2(Z_Y)} \right] \\
r_3(X = Z_Y) &= \sum_{Z_Y=\neg Y\cap Z} \left[\frac{n_1(C) \cdot n_2(Z_Y)^2}{n_1(C) + n_2(Z_Y)} + \frac{n_1(Y_Z) \cdot n_2(Z_Y)^2}{n_1(Y_Z) + n_2(Z_Y)} \right]
\end{aligned} \tag{5}$$

and

$$\begin{aligned}
n_1(C) &= \alpha_1 \cdot m_1(Y) \quad \text{and} \quad n_1(Y_Z) = (1 - \alpha_1) \cdot m_1(Y) \\
n_2(C) &= \alpha_2 \cdot m_2(Z) \quad \text{and} \quad n_2(Z_Y) = (1 - \alpha_2) \cdot m_2(Z)
\end{aligned} \tag{6}$$

In Formula 6, n_1 and n_2 perform an even distribution of the belief mass between C and Y_Z, and C and Z_Y; if $C = \emptyset$ then $\alpha_1 = \alpha_2 = 0$, if $Y_Z = \emptyset$ or $Z_Y = \emptyset$ then $\alpha_1 = 1$ or $\alpha_2 = 1$ respectively, otherwise $\alpha_1 = \alpha_2 = \frac{1}{2}$.

7 Related work

Much research has been done in how software agents can be designed to effectively support humans teams (e.g. [1, 15, 4] to name a few), but most of this work involves human participants interacting with software agents through interfaces that represented a simplified operational environment (e.g. 2D maps, switches and buttons on the screen), and where every single parameter governing the simulation could be monitored and controlled by the experimenters (no uncertainty). In general participants interacted with the interface in a point-and-click fashion or using vocal commands, but there was no physical action simulation.

On the other hand, in a number of domains such as military operations or emergency response, human teams are physically involved in the plan execution while a number of decisions needs to be made, for example, because of unexpected situations coming along or because a contingency course of action needs to be initiated. As a consequence, being already cognitively committed to perform an activity, they are less willing to spend time in processing new information, especially if it contradicts their actual beliefs about the plan development (known also as confirmation bias [7]). We are thus following an approach similar to that reported in [14, 6], where a first-person computer game is employed (and possibly adapted) in order for human teams to act in more realistic setting that reflects better the scenarios in which both humans and software agents will eventually operate together. This approach may present some shortcomings, because, as there are no real-life consequences from their actions, participants might not behave exactly how they would in a real-world exercise, sometimes becoming bored or reckless [6], but it is in our opinion more realistic than point-and-click interfaces.

8 Conclusions

This paper presents research in progress concerning monitoring and synchronising human teams during the enactment of a plan of action in an uncertain environment, by supporting them through software agents. The design of such agents is not an easy task, because it requires them to be aware of what is happening in the environment in a way similar to what humans do, however we believe that by combining behavior recognition techniques and evidentail reasoning, this can be achieved. We have proposed a way in which a plan can be encoded into the agent design and the minimal amount of knowledge the agent needs to gather both during the design and the execution phases in order to effectively monitor the enactment. Finally we have outlined a three-step approach to our design. We are now working towards an implementation and we expect to test our implementation using a first-person game as a realistic simulation environment.

References

1. J. Allen and G. Ferguson. Human-Machine Collaborative Planning. In *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*, October 2002.
2. M. Bauer. A Dempster-Shafer approach to modeling agent preferences for plan recognition. User Modeling and User-Adapted Interaction. *User Modeling and User-Adapted Interaction*, pages 317–348, 1995.
3. N. J. Blaylock. *Towards tractable agent-based dialogue.* PhD thesis, Department of Computer Science, University of Rochester, August 2005.
4. X. Fan, S. Sun, M. McNeese, and J. Yen. Extending the Recognition-Primed Decision Model to Support Human-Agent Collaboration. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.
5. X. Fan and J. Yen. Realistic Cognitive Load Modeling for Enhancing Shared Mental Models in Human-Agent Collaboration. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.
6. D. Masato, T. J. Norman, S. Poltrock, H. Bowyer, and P. Waggett. Adaptive Military Behaviour in a Collaborative Simulation. In *SISO European Simulation Interoperability Workshop*, July 2008.
7. R. S. Nickerson. Confirmation Bias: A Ubiquitous Phenomenon in Many Guises. *Review of General Psychology*, 2:175–220, 1998.
8. S. Poltrock, M. Handel, H. Bowyer, and P. Waggetg. A Dynamic Model of Mission Context. In *Procceedings of the Second Annual Conference of the International Technology Alliance in Network and Information Science*, September 2008.
9. S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach.* Prentice Hall, Upper Saddle River, New Jersey, USA, second edition, 2003.
10. K. Sentz and S. Ferson. Combination of Evidence in Dempter-Shafer Theory. Technical Report SAND2002–0835, Sandia National Laboratories, 2002.
11. G. Shafer. *A mathematical theory of evidence.* Princeton University Press, Princeton, New Jersey, USA, 1976.
12. E. Sirin and B. Parsia. Planning for Semantic Web Services. In *Semantic Web Services Workshop at the Third International Semantic Web Conference*, 2004.
13. G. Sukthankar and K. Sycara. Simultaneous Team Assignment and Behavior Recognition from Spatio-temporal Agent Traces. In *Proceedings of Twenty-First National Conference on Artificial Intelligence*, July 2006.
14. G. Sukthankar, K. Sycara, J. A. Giampapa, C. Burnett, and A. Preece. Towards a Model of Agent-Assisted Team Search. In *Procceedings of the First Annual Conference of the International Technology Alliance in Network and Information Science*, September 2007.
15. K. Sycara and M. Lewis. *Team Cognition: Understanding the Factors That Drive Process and Performance*, chapter 10, pages 203–233. American Psychological Association, 2004.
16. D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S Web Services Composition using SHOP2. In *Proceedings of the Second International Semantic Web Conference*, October 2003.
17. K. Yamada. A New Combination of Evidence Based on Compromise. *Fuzzy Sets and Systems*, 159(13):1689–1708, 2008.

Learning User Characteristics for Efficient Interruptability in Agent–User Collaborative Systems

Tammar Shrot[1], Avi Rosenfeld[2], and Sarit Kraus[1]

[1] Bar-Ilan University, Ramat-Gan 52900, Israel
{machnet, sarit}@cs.biu.ac.il
[2] Jerusalem College of Technology, Jerusalem 91160, Israel
rosenfa@jct.ac.il

Abstract. In teamwork when a user and an agent are working together on a joint task it may be important to share information in order to determine the appropriate course of action. However, communication between agents and users can constitute costly user interruptions. One of the most important issue concerning the initiation of information sharing in teamwork is the ability to accurately estimate the cost and benefit arising from those interruptions. While cost estimation of interruptions has been investigated in prior works, all of those works assumed either a large amount of information existed about each user, or only a small number of states needed consideration. This paper presents a novel synthesis between Collaborative Filtering methods together with classification algorithms tools in order to create a fast learning algorithm. This algorithm exploits the similarities between users in order to learn from known users to new but similar users and therefore demands less information on each user. Experimental results indicate the algorithm significantly improves system performance even with a small amount of data on each user.

Key words: Cost Estimation, Collaborative Filtering, Classification Algorithm

1 Introduction

An important aspect of teamwork is efficient inner-team communication [1]. When working together on a joint task it is important to share information in order to coordinate consequent actions. In addition, different agents on a team often possess information required by others. The need for efficient communication arises in mixed human-computer teams as well as in homogeneous computer-agent environments [10]. However, it is important to appropriately time the communication, since poorly timed interruptions can lead to averse effects on task performance and on a human agent's emotional state [1].

Cost estimation of interruptions has been investigated in prior works (see Section 2.1). However, these methods require many repetitive interactions or a very strict domain with a small number of states. In the domain we studied,

as well as in many other domains, we assume that these repetitive iterations incur a very high cost. In addition, some applications, such as on–line bidding agents have a limited number of iterations with each user making this approach impractical.

We focus on developing novel algorithms that can quickly and accurately model user preferences. Our research was specifically motivated by DARPA's Coordinators program[3]. The basic goal of this program is to create automated decision support units that better focus a user's attention in dynamic environments. We consider a teamwork framework where agents and human operators work together on a joint task and have unique capabilities. Human operators are assumed to have access to more complete domain knowledge or expert knowledge external to their agents. However, people's time is valuable, and thus the team incurs a cost every time the agent interrupts the person to obtain the information she may have. The profile of the specific user receiving the query can quickly change as the user's availability to provide information is subject to dynamics based on the person's ability to be interrupted, her current activity or the current state of the environment. The agent must decide if it should initiate a query to a human operator based on evaluating whether the cost of interrupting the user is outweighed by the value gained from her knowledge.

The basis of our solution is the assumption that users can be clustered such that the behavior of one user will be similar to the other users in her cluster. Thus, once we have knowledge about some users we can generally estimate the value and cost associated with an agent–human interaction of new, but similar users as well. Specifically, we propose the use of a new user modeling approach with elements of Collaborative Filtering (CF) algorithms. Traditional CF algorithms are typically applied to recommend a given product (book, movie, game, etc.) to a user based on information gleaned about general users' buying behavior. Collaborate Filtering analyzes the relationships between users and interdependencies among products, in order to identify new user–item associations. In addition, to avoid the need for extensive data collection about items or users, Collaborative Filtering requires no domain knowledge [3].

We propose a new approach of combining components of Collaborate Filtering algorithms with basic classification algorithms such as the *C4.5 Decision Tree* algorithm [15] or the *k-nearest neighbor* ($k-NN$) algorithm [5]. The advantage of this synthesis over traditional learning methods is its significant reduction in the learning time needed to model a given user. This allows us to quickly decide about the efficiency of an interruption with only limited data and can avoid pitfalls such as protracted learning periods and elicitation of private user data. Our approach is also significantly distinguished from traditional Collaborate Filtering algorithms and Machine learning algorithms. Three major differences exist between our approach and Collaborate Filtering algorithms. First, in Collaborative Filtering algorithms the similarity between users is decided by shared habits, and the similarity between items is decided by shared history. In our domain such shared information does not exist and other, machine

[3] http://www.darpa.mil/ipto/programs/coor/coor.asp

learning, tools are used. Second, several traditional Collaborate Filtering algorithms work by identifying the precise value of a current parameter state (e.g. the genre of a movie) and use this information to decide whether to recommend an item [7]. In our approach, we allow a range of possible parameter values. The result is an algorithm that can effectively make decisions without precise state information. Third, our approach has no need for personal information (demographic information, economic and marital status, age, etc.) typically used in Collaborate Filtering algorithms [7]. Many users are reluctant to divulge private information, leading to many people refraining from these systems. In contrast, our model focuses exclusively on the user's interaction with the agent, and thus does not require sharing any personal information. In addition, our approach significantly distinguished from machine learning algorithms. In contract two machine learning algorithms, our approach uses two different phases to decide about the situations' similarity level and each phase uses different type of data regarding the situation.

2 Related Work

We propose a new method for cost estimation with is motivated by Collaborative Filtering methods.

2.1 Cost Estimation of Interruptions

One of the most important issues concerning the initiation of teamwork interruptions is the ability to accurately estimate the cost and benefit arising from the interruption. Accurate estimation will enable interruption only when it will have a positive ipmact on the team's performance.

Previous works have investigated how to estimate the cost of interruptions [8, 18]. Fleming and Cohen [6] were the first to build a user-specific model which generally takes the user's specific factors into account. They used cost estimation to create a decision making mechanism in order to decide when to initiate communication. Horvitz and Apacible [8] studied the user's benefit from receiving information from the computer agent, while our study focus on the opposite situation — the agent wishes to receive information from the user. Tambe et al. [18] focused on when to turn control to the user, instead of information transfer. Sarne and Grosz [16] offered an efficient model to manage the agent's information and to decide when to interrupt the user. Kamar et al. [10] used POMDP and MDP models to evaluate the cost of interruption while taking into account the possible mismatch between the computer's calculation of utility and the person's perception of it. The key difference in our research is we study cost-estimations that can work in dynamic domains in which the environment's conditions rapidly change, actions occur quickly, user's abilities change over time and decisions must be made within tightly constrained time frames. For example, Nonetheless, work by Sarne and Grosz [16] needs a large amount of information on each user before it can begin to operate at an efficient level. Work by Kamar et al. [10] considers a

limited domain with a small number of states and is computationally–infeasible for domains with many dynamic states.

2.2 Collaborative Filtering

Collaborative Filtering (CF) is a method of making automatic predictions (filtering) about the preferences of a user by collecting data on the preferences of many users (collaborating). There are hundreds of examples of recommendation systems via Collaborative Filtering. Surveys of recommendation systems and collaborative systems are presented in [20, 4, 7]. User profiling and matching mechanisms are illustrated, especially on Collaborative Filtering techniques. In addition, a number of Collaborative Filtering algorithms are compared for accuracy of available test data.

Collaborate Filtering models can be built based on users or items. User Based collaborative filtering systems find other users that have displayed similar tastes to the active user and recommend the items similar users have preferred [2, 12, 14]. Item Based models recommend items that are most similar to the set of items the active user has rated [9, 17, 13].

Karypis [11] was the first to recommend an approach that combines the best of the Item Based and the User Based (classic Collaborate Filtering) algorithms, by first identifying a reasonably large neighborhood of similar users and then using this subset to derive the Item Based recommendation model. Vozalis et al. [19] have developed a hybrid method consisting of a number of steps. In the first step, the algorithm creates a neighborhood of users most similar to the selected active user; it first calculates the similarity level between the users and the active user and then chooses the k-nearest neighbor who rated a large number of items. In the second step, the rating of the k users is used to calculate item-similarity between these users. In the final step, the algorithm recommends the active user items similar to the items chosen (based on the first two steps).

In our research we leverage these approaches to quickly learn about new users from known users. We present a novel variation of traditional learning algorithms that applies the tools and principles defined in the hybrid User and Item Based Collaborative Filtering method in order to incorporate them into our learning algorithm. In the next section, we present specifics of the Coordinators Domain we studied, as well as specifics of our algorithms.

3 A Synthesis between CF and Machine Learning

We generally model the Coordinator's joint agent-human teamwork domain problem as follows: Assume a user has to complete a task within a limited time. The task consists of many sub-tasks. Each successful completion of a sub-task entitles the team to a certain gain, referred to as "*taskGain*", different types of task have different gain value. In addition, successfully transferring desired user information to the agent entitles the team to a certain different gain, referred to as "*comGain*", this value changes according to the information accuracy.

1. Accept a new non labeled situation $s = (i, u)$.
2. Use u to create a user profile p for s.
3. Use the profile p to build a neighborhood NGB of users that were found to be similar to u.
4. $Items = \emptyset$
5. $\forall s' = (i', u')\ s.t.\ u' \in NGB \quad Items = Items \bigcup \{i'\}$
6. Build a *classification* model CM between i and $Items$ using a classification algorithm.
7. Label the new situation s according to the classification decision of CM.

Fig. 1. Algorithm for deciding whether the timing is good or bad.

Our proposed algorithm (Figure 1) is motivated by the Collaborative Filtering hybrid approach. Just like in the Collaborative Filtering algorithms our method has two phases: a "user" phase, and a "item" phase. The first phase uses user specific data in order to identify similar users in the database and construct an environment of similar users. The second phase of the algorithm uses state specific data in order to decide about the state. However, the second phase only takes its information from states that belong to users in the "similar environment" built in the first phase (defined herein).

Each data point in our database (situation) is constructed from two distinct data types: user latest history and interruption profile. User latest history is a collection of vectors with information gathered within a short period of time (20 seconds in our experiments) before the agent considers if it should interrupt the user. This historical information shows how the user behaves and enables the algorithm to construct a profile of the user's behavior. The second data type - the interruption profile is a vector that describes the user's state immediately prior to the time of the proposed interruption. Each of these vectors (from the user's profile and interruption profile) contains numerical and/or other discrete values for different attributes. The users' history and interruption profiles vectors' attributes, include for example information regarding the user's location, percentage of the task accomplished, time from the beginning of the task, location of the nearest disturbing elements, etc. All values in all vectors' attributes are normalized to the same scale.

While the algorithm's structure is motivated by Collaborative Filtering methods, the tools actually used to decide the user's neighborhood (similar users) and if an interruption should take place are machine learning tools. Collaborative Filtering methods cannot feasibly be implemented on the given data. In teamwork domains, unlike in Collaborative Filtering, users cannot be compared according to shared habits and items (states) cannot be compared according to shared history. Consequently, different methods must be found to model new users's and items's similarities. Since our data is labeled, our solution is to use traditional machine learning classification algorithms. The classification algorithms are used to quickly compare the users (in the first phase) and items (in the second phase) without resorting to a shared habits or shared history.

In the first phase ("user" phase) the algorithm builds a "user's" similarity model between the new given situation and the given labeled situations in the database (Lines 2 and 3 in the algorithm). This phase uses only the first data type in the situation (user's latest history). The users' similarity model is built according to the similarity between the "user latest history" data type in the situations. Specifically, the user's latest history is a set of x vectors that represents the user's behavior in the short period sampled before the interruption (20 seconds in our experiments). For each user, the algorithm calculates the changes in the user's attributes value throughout the latest history measuring time. Then, for each attribute it is calculating the average change in the user's attributes values (Line 2 in the algorithm). The user's profile is the vector of averages changes. Namely, the algorithm uses the user's latest history in order to calculate the average change in the user behavior (values) in this time sequence. Then, the similarity between two users is measured as the distance between the two calculated profiles (Line 3 in the algorithm). The assumption behind this approach is that users with similar profiles (same average change in values) undergo the same process and therefore most probably act in similar ways. This model represents the user's similarity level between the new situation and the given labeled situations. Once the similarity model is complete, the algorithm takes the l situations of the most similar users in the database as the new situation's neighborhood, and uses this neighborhood in the algorithm's next stage. Notice, that this phase is using tools taken from the *k-nearest neighbor* $(k-NN)$ algorithm [5] in order to locate the new situation's neighborhood.

The next stage of the algorithm (Lines 4 to 7 in the algorithm) uses only the second date type in the situation (interruption profile). Once the user's profile and neighborhood are constructed, the next stage of the algorithm is to build an "item's" (interruption profiles) similarity model between the situations that belong in the neighborhood. Once the above algorithm identifies the user's neighborhood in the first phase it uses a machine learning classification algorithm on the items that belongs to the neighborhood's users and returns the calculated classification. The net result is that once a new situation arrives, the need only a very short time to gather enough data in order to decide how to treat it. This allows for a faster and more accurate classification than the base machine learning algorithms alone could provide. While our approach is meant to work with any machine learning algorithms, we specifically considered two classification algorithms: the *C4.5 Decision Tree* algorithm [15] and the *k-nearest neighbor* $(k-NN)$ algorithm [5].

4 Experimental Results

In order to study our new algorithm and its ability we conducted a number of experiments. This section contains the experiments we did and the rational behind them, as well as the description of the environment used for our experiments.

4.1 Experimental Environment

We are currently conducting experiments with a game setting called "Final Frontier Trader"[4]. The goal of the game is to destroy all enemy ships and asteroids. For every asteroid or ship destroyed the team earns a certain gain. We refer to the constant score obtained from destroying an asteroid as *astGain*, and the constant gain from destroying a ship as *shipGain*. The game has a time limit (ten minutes) and the human operator (user) must destroy all enemy ships and asteroids before the end of the game. A user who dies during the game (e.g. shot by enemy ships) or the time limit passed loses a large number of points. On the other hand, the faster the user accomplishes her main goal the larger the gain she earns from successful task completion. Additionally, the user can increase the team's gain by answering questions from the agent (*comGain*). We assume that while the agent asks a question, the user cannot simultaneously perform her previous subtask. Thus, while the user is prompted for information the game continues, the user's ship continues moving, and enemy ships can potentially shoot at and even destroy the user's ship. Control is returned to the user only after she correctly answers the question. Moreover, for each wrong answer the user provides, the value of *comGain* rewarded to the team is reduced. While the algorithm was designed for an agent–user collaborative system (mixed–initiative system), the environment used to test it was not mix–initiative environment. This is due to the fact that in this stage of the research we simply want to make sure the algorithm is effective. Therefore, the agent's questions are simple mathematical questions which are presented as multiple choice (*SAT*) questions. We assume that these questions may hurt the human's performance. Our goal is for the agent to ask a given question only when the cost to the user for answering the question will be lower than the gain from answering the question.

The agent collects information regarding the user's state in regular intervals (life status, location, percentage of the task accomplished, time from the beginning of the task, location of nearest enemies, etc.). This information is necessary to construct the user's state model used by our algorithms.

In our research protocol each subject played three games:

1. A simple scenario with simple questions – users' game learning session.
2. A complex scenario without questions – experiments without question.
3. The complex scenario with questions – the experiment session.

All subjects played scenario 1 first, as a training session about the game's controls. The order in which they played scenarios 2 and 3 varied. 50% of the users played the experimental scenario without questions before playing it with questions, and 50% played it with questions before playing it without questions. This was done to negate any impact on results from the order these scenarios were played in.

The information gathered from scenario 1 was omitted from the analysis of the results, since these games were only used to teach the users about the environment. The information gathered from scenario 2 was compared with the

[4] http://fftrader.sourceforge.net

information from scenario 3 in order to isolate the effect the questions had on the user performances and make sure that they interrupted the users. This information was used while labeling the questions in the database. Scenario 3 was the experimental scenario used to evaluate our algorithm.

Each subject performed the entire experimental protocol as described above. The subject's state and status were sampled every 5 seconds (user's latest history) and also before (interruption profile) and after each interruption in the scenarios with questions. At the end of the research protocol each question (interruption) was labeled as either "good" or "bad" according to the effects it had on the score and according to the amount of interruption it caused the user (the amount of life lost, if the user was set out off course, etc.). The labels were determined based on an analysis of the difference between the "before" and "after" information as well as the final score the user achieved in the game.

We ran a 10–fold cross–validation test on the data collected from the third scenario. In each of these 10 iterations a subset of the data (90%) was randomly chosen, and used in order to build the model. The rest of the data was used to test the model. The situations from the test data were inserted into the model as new situations that must be analyzed. The model decisions were compared to the situations' labels. Four decision models were examined:

- A naive $k - NN$ classification model.
- A naive $C4.5$ decision tree classification model.
- A "User Based" + $k - NN$ model ($UB + KNN$).
- A "User Based" + $C4.5$ decision tree model ($UB + C4.5$).

4.2 Testing the Algorithms

We posit that our algorithm can learn from similar subjects to new subjects and correctly decide whether it is currently a good or a bad time to interrupt the subject. To test this hypothesis, we ran the above research protocol on a group of 56 people who varied in age, occupation and level of computer knowledge. Each person had between 4 to 25 interruptions in his game (average value of 20 interruptions per game). Therefore, we had approximately 1120 situations (user's latest history + interruption's profile) in our database. For each situation we had a short history from the user's activity in the previous 20 seconds (about 4 - 5 vectors) which was used to identify users' similarities, and an interruption's profiles used to identify items' (states') similarity. The scenarios used have a large number of enemies and their aggressive level varied from low to high. Therefore, the number of possible states the users could encounter was enormous. We deliberately studied a large variety of different users and states so we can examine the true influence of the "User Based" approach.

The results, as presented in Figure 2, support our claims. They show that in a heterogeneous environments users using the "User Based" approach significantly improves the system performances. We can see that there is a significant change ($p < 0.01$) between the naive algorithms and the enhanced "User Based" algorithms. In addition, it is seem that there is no significant difference ($p > 0.05$)

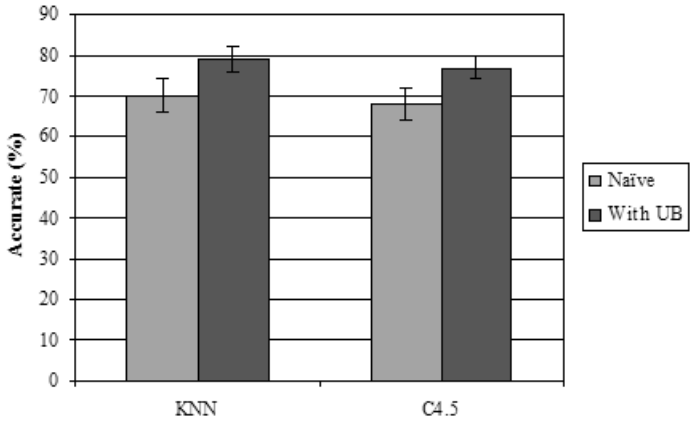


Fig. 2. The algorithm accuracy percentage as a function of the different algorithms.

between the naive $C4.5$ decision tree algorithm and the naive k-nearest neighbor ($k-NN$) classification algorithm. It is interesting that the both naive algorithms ($C4.5$ and $k-NN$ algorithms) were improved by a very similar factor with no significant difference ($p > 0.05$) between "User Based" enhance $k-NN$ and the "User Based" enhance $C4.5$ algorithms. It shows that both algorithms can significantly benefit from using the "User Based" approach in heterogeneous environments.

4.3 The Effect of Adding Users to the System

All the algorithms discussed in this paper assume that there is an initial labeled database that can be used in the classification algorithms. However, questions arise as to how much initial training is needed before the system can begin to be used. Therefore, we studied how the size of the initial database effects each of the algorithms, and attempted to reveal the minimal amount of training data needed for each algorithm. During this experiment it was important to make sure that while enlarging the number of users, we are not changing the level of heterogeneity among the users. This was done by using the procedure described in section 4.4.

In order to simulate a larger database we enlarge the number of subjects used in the experiments. On average, each subject added 20 new situations to the database. Namely, when we have 30 subjects in the experiment we have 600 situations in the database.

Several interesting phenomenon appear in Figure 3. First, as stated in Section 4.2, using "User Based" aspects from Collaborative Filtering improves both naive machine learning algorithms by the same factor. This significant improvement is found even in a small database that contain 20 subjects ($\sim$ 400 situations).

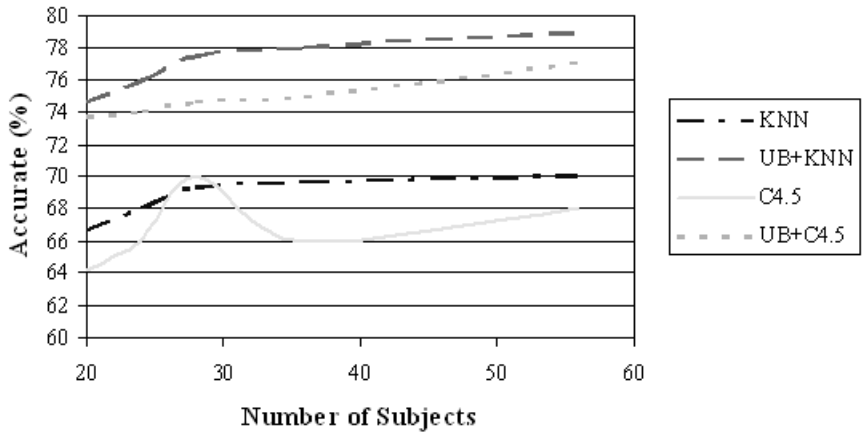


Fig. 3. The algorithms accurate percentage as a function of the number of subjects.

Second, it seems that while the number of users was smaller than 30 (around 600 situations in the database) the algorithms were not stable. However, when the number of users exceeds 30 (more than 600 situations) all algorithms became stable, including the our algorithms that use less data in order to build their model ($l = |database| * 0.1 \approx 60$ situations). Apparently, the similarity between users is strong enough to achieve a significant improvement over the naive algorithm even with a small amount of data.

4.4 The Effect of Changing the Heterogeneity Level of the Subjects

One of our hypothesis is that as the users' level of heterogeneity increases the larger the gain we have from using the "User Based" approach. This is because the Collaborative Filtering "User Based" approach learns based on user differences.

To understand differences between users, we divided them into the following four groups: the "gamer" group; the "skilled" group; the "fair" group; and the "hopeless" group. We asked user's to self-describe which group they believed they would fall in. Subjects that their self definition did not match their performances were removed from all our experiments, leaving us with 56 subjects.

The number of situations were kept constant during this experiment, and the "subjects' level of heterogeneous" was modified by controlling the ratio of subjects that came from each group.

The results in Figure 4 show that the "User Based" approach performs better in heterogeneous environments. We define that heterogeneous level means the ratio between the groups in the general population. Namely, heterogeneity of

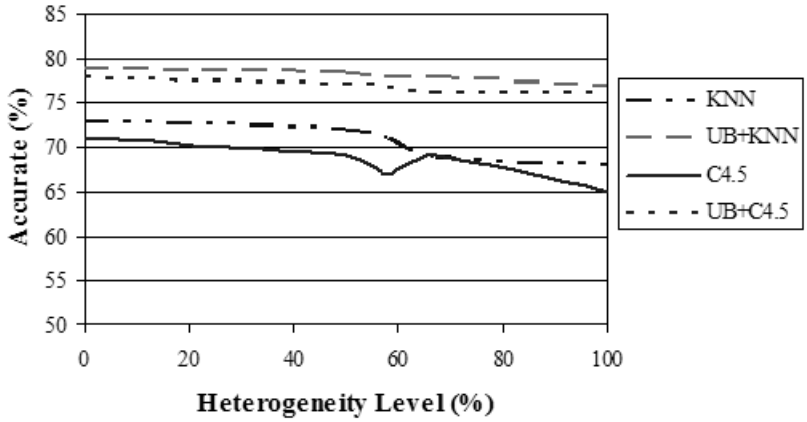


Fig. 4. The algorithms accurate percentage as a function of the heterogeneity's level.

0% means that all situations in the database are from the same group; heterogeneity of 100% means that each group has an equal number of situations in the database. Notice, that while the "User Based" algorithms are negligibly effected by the reduction in the user's heterogeneity level, the naive algorithms perform much better. Therefore, as the heterogeneity level decreases the "User Based" approach shows less improvement compared to the naive algorithms.

This result is not surprising. As the heterogeneity level in the database increases, the learning task becomes harder as more problem instances are increasingly less similar. Therefore, it is obvious that naive classification algorithm will produce less suitable results.

5 Conclusions and Future Work

This paper introduces a novel approach to combine Collaborative Filtering methods with classification algorithms tools in order to create a new fast user characteristic algorithm for mix agent–user system. This approach significantly improves system performances even in the absence of sufficient information for learning or user modeling. We found that after a small initial database was created with as little as sparse data from 30 subjects, our algorithms were able to accurately model the subject's interruption preferences with nearly 75% success in less than 20 seconds!

We conclude that the algorithms we present can provide a good solution for semi–tailored applications that have a large number of users but only a short dialog with each user. These applications cannot realistically gather enough information on a single user in order to grant her the personalized service she requires. However, an application can use the offered approach in order to learn from past users and quickly profile new users.

There are many directions we would like to pursue in our future work. First and foremost, we would like to preform "on-line" experiments. The experiments preformed in this paper used an "off-line" experimental procedural. Namely, the subjects were given questions in fixed time intervals, and the analysis on the data using our new algorithm was done off-line. We would like to conduct experiments where the decision whether to ask a question in a given time will be decided on-line using our proposed algorithm. Another direction for future research is to investigate other domains and problems, especially mix–initiative problems. For example, we hope to study situations where providing information to agents does not directly increase the team's value, but rather provides the team with additional abilities that would enable it to perform new tasks, or improve their performance in existing tasks. Similarly, we currently studied user-agent interruption manager only as modeled through answering mathematical questions. We would like to examine different types of interruptions, i.e., in a mix–initiative environment where the user and agent share the same goal and the interruptions are questions regarding common actions. In addition, we would like to investigate the influence causes by the measuring time duration of the user's latest history. We would like to find out if longer measuring time will enable better representation of the user's profile. Finally, interesting questions may address what is the sufficient level of model accuracy needed to improve performance. Is the 75% level of accuracy our model achieved in 20 seconds sufficient, or should the aim be for a more accurate model, but with a longer training period? We are confident that the novel synthesis of Collaborative Filtering and traditional learning methods presented in this paper will stimulate interesting innovations in the future.

References

1. P. Adamczyk and B Bailey. If not now, when?: the effects of interruption at different moments within task execution. *In CHI 04*, pages 271–278, 2004.
2. Charu C. Aggarwal, Joel L. Wolf, Kun lung Wu, and Philip S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 201–212. ACM Press, 1999.
3. R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
4. Breese, Heckermen, and Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Technical report, Microsoft Research*, 1998.
5. B.V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Computer Society Press, 1991.
6. M. Fleming and R. Cohen. A user modeling approach to determining system initiative in mixed-initiative ai systems. *In UM 01*, pages 54–63, 2001.
7. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM Press.

8. E. Horvitz and J. Apacible. Learning and reasoning about interruption. *In ICMI 03*, pages 20–27, 2003.
9. C.-S. Hwang and P.-J. Tsai. A collaborative recommender system based on user association clusters. *Lecture Note on Computer Science*, vol. 3806:463–469, 2005.
10. Ece Kamar, Barbara J. Grosz, and David Sarne. Modeling user perception of interaction opportunities in collaborative human-computer settings. In *AAAI 2007*, pages 1872–1873, 2007.
11. George Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, pages 247–254, 2001.
12. R. Lin, S. Kraus, and J. Tew. Osgs - a personalized online store for e-commerce environments. *Information Retrieval journal*, vol.7(3–4):369–394, 2004.
13. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing 7*, pages 76–80, 2003.
14. S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS) 22(1)*, pages 54–88, 2004.
15. John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
16. D. Sarne and B. J. Grosz. Timing interruptions for better human-computer coordinated planning. *In AAAI Spring Symp. on Distributed Plan and Schedule Management*, 2006.
17. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *Proc. 10th International Conference on the World Wide Web*, pages 285–295, 2001.
18. M. Tambe, E. Bowring, J. Pearce, P. Varakantham, P. Scerri, and F. Pynadath. Electric elves: What went wrong and why. *In AI Magazine*, 2007.
19. M. Vozalis and K. G. Margaritis. On the combination of collaborative and item-based filtering. In *presented at the 3rd Hellenic Conference on Artificial Intelligence (SETN 04)*, Samos, Greece, 2004.
20. Jiangshan Xu, Liang-Jie Zhang, Haiming Lu, and Yanda Li. The development and prospect of personalized tv program recommendation systems. In *IEEE Fourth International Symposium on Multimedia Software Engineering (MSE'02)*, pages 82–89, 2002.

The Effects of Cooperative Agent Behavior on Human Cooperativeness

Arlette van Wissen, Jurriaan van Diggelen, and Virginia Dignum

Institute of Information and Computing Sciences,
Utrecht University, the Netherlands
arlette.vanwissen@phil.uu.nl; {jurriaan,virginia}@cs.uu.nl

Abstract. Within negotiation environments, cooperative behavior may emerge from different factors. In this paper, we focus on human-agent interaction and in particular on the question of how the cooperativeness of a software agent affects the cooperativeness of a human player. We implemented three different kinds of agent behavior to determine how they influence helpful human behavior. Our data shows that humans behave more cooperatively towards agents that negotiate with them in a cooperative way and that humans tend to punish egoistic and unfair play by behaving non-cooperatively themselves.

Keywords: human-agent interaction, negotiation, cooperation, Ultimatum Game

1 Introduction

Social relations between software agents and humans is becoming an essential part of our life. Both in everyday life and science humans use, control and cooperate with agents [1]. Many forms of human-machine interaction involve participants that pursue different, sometimes even contradictory, interests. In these scenarios, humans and agents are autonomous entities which interact and cooperate with each other while keeping in mind their own objectives and goals. In this work, we use the word 'agents' to refer solely to software agents and 'actors' to refer to both agents and humans. The interactions between these actors take place in many domains such as games [2], where humans play against agents with contradictory goals, and online auctions [3], where humans and agents compete over the same product. To achieve successful human-machine interaction, we must understand the factors involved in this interaction.

The area of human-agent interaction has long been neglected, ignoring the issue of how the interaction affects the behavior and strategies of the involved actors. Most work concerned with human-agent interaction focuses mainly on how an agent should be designed to model its opponent or on how to design most efficient agent strategies. Still only a very limited number of studies shed light on the social phenomena that occur between humans and agents and on

the influence of this interaction on human behavior [4–6].

This work presents an experimental study of the social phenomena that play a role in human-agent interactions with contradictory interests. In this project we combine and extend existing work to understand the effect of agent behavior on the cooperativeness of people towards their opponents. More specifically, we explore to what extent social principles like reciprocity and fairness influence human bidding behavior in these interactions. This paper is a more elaborate description of the work we presented in [7]. The central question we address is: how does the cooperativeness of an agent affect the cooperativeness of a human player? The study of human bidding behavior in relation to agents with competing goals may provide new insights into how bidding decisions are made and how agents' behaviors are perceived. This knowledge helps to establish successful human-agent interaction.

Since our work concentrates on the interaction between humans and agents, our research method combines methods and results from both behavioral and computer sciences. On the one hand we study human behavior by means of experimental research. We build upon earlier research showing that human reasoning and negotiation strategies are affected by social factors, such as altruism, fairness and reciprocity [4, 8]. On the other hand, we use observations from game theory of cooperation between agents and succesful reciprocal agent strategies. A mixture of punishing defection and rewarding cooperation seems to encourage cooperation [9]. It has also been shown that a significant difference between the reaction of people to agent and to human opponents exists [5, 10]. We combined these observations to test two main hypotheses. First, humans will behave cooperatively towards an altruistic opponent. Second, they will punish egoistic behavior. These hypotheses are discussed in more detail in Section 3.3. Both hypotheses assume similar behavior towards the human and agent opponents and therefore that social tendencies predominate over any prejudices about the nature of the opponent.

To analyze differences in behavior between various kinds of human-agent interactions we use Colored Trails (CT) [11], which is a negotiation environment for designing and evaluating decision-making behavior and dynamics between players. CT is a conceptually simple but strategically complex game that has been developed for testing strategies for automated agents that operate in groups. In our experiment, mixed groups of humans and agents play the game. We implement different agent strategies that vary in their degree of cooperativeness.

We create a CT setup that enables multiple humans to play the game simultaneously against alternate opponents. In some conditions of the experiment subjects were not aware of the identity of their opponent, i.e. they did not know whether they were playing against a human or an agent. This enables us to analyze the relation between the cooperativeness of agents and that of humans.

Our results support our hypotheses and show three main findings. First, humans tend to be more cooperative towards a cooperative opponent and do not fully exploit altruistic behavior. Second, humans play less cooperatively towards egoistic opponents. Third, an egoistic opponent is often perceived as a human.

The paper is organized as follows. First, we discuss related work and show how our work differs from it. The conceptual approach to our experiment is presented in Section 3. This section is mainly concerned with introducing the Colored Trails game framework and demonstrating how we use it in this project. In Section 4 the experimental setting will be addressed. We will discuss the results of the experiments in Section 5, followed by a brief conclusion and outline for future work in Section 6.

2 Related Work

This paper may be placed at the intersection of studies on human negotiation behavior and studies on agent negotiation behavior. It therefore builds on three streams of research: human-human interaction, agent-agent interaction and human-agent interaction. These fields are briefly discussed in the following three subsections.

2.1 Human-Human Interaction

The Ultimatum Game [12] is a commonly used game setting to simulate and analyze negotiation behavior that was originally developed by experimental economists. In the classic ultimatum game, two people are given the task to divide a sum of money. One player proposes how to divide the money between them and the other player can accept or reject this proposal. If the second player rejects, neither player receives anything. In the original game, the players interact only once. As it turns out, people usually offer 'fair' (i.e., 50:50) splits, and most offers of less than 20% are rejected. This latter phenomenon is referred to as *inequity aversion*.

Many experiments have been conducted that confirm these properties of human behavior. Social factors, such as altruism, fairness and reciprocity are shown to have an influence on negotiation behavior [8, 13]. When playing against other humans, humans tend to be cooperative towards their opponents but punish them severely in case of unfairness. As soon as human actors are involved, people seem to develop expectations of how other people should behave. People not only care about the outcome of actions; they are also concerned with how they come about [14]. These results show that humans do not always pursue their ultimate utility. Apparently, humans are not completely rational actors as assumed in standard economic game theory [15]. In our study, we use these results to formulate the hypothesis that human expectations of an opponent are also applicable to agents.

2.2 Agent-Agent Interaction

Negotiation between intelligent agents has been widely studied over the past years. Mostly, a Multi-Agent System (MAS) is assumed to consist solely of computer agents. Interacting agents are faced with the challenge of modeling other

agents and their behavior in order to adapt to their environment [16]. Much research has been done on social reasoning models for agents. These models, as opposed to the models which are fully based on game theory, may enable the agent to be more adaptive to the environment and to perform better in more realistic negotiation scenarios [17]. Axelrod [9] has demonstrated that in some MAS's, it can be beneficial for agents to use a cooperative strategy. In these situations cooperative agents do as well as or outperform competitive agents. Moreover, reciprocal behavior allows cooperators to do better than self-interested rational agents [18]. Models that take social principles such as fairness and helpfulness into account were shown to explore new negotiation opportunities [19] and find solutions that correspond to solutions found by humans [17].

2.3 Human-Agent Interaction

Our study is based on two seemingly contradictory findings from empirical research. On the one hand, people seem to display the same social behavior towards agents as they do to humans. The negotiation experiments presented in [4] seem to imply that human responders still care about equality of outcomes while negotiating with agents. On the other hand, experiments indicate that a significant difference exists between the behavior of humans towards agent and human opponents. Humans seem to perceive other humans differently from agents. For example, experiments by Sanfey et al. [5] and Blount [10] show that humans are more likely to accept a proposal from agents than from other humans. More seems to be tolerated in terms of behavior and actions from agent actors than from human actors. Humans and agents seem to have a different set of 'acceptable behavior'. This leaves us with the question how humans actively treat agents in a negotiation setting: Are social factors like fairness and altruism less applicable to agents since humans do not ascribe the same expectations to them?

None of these prior works have investigated the effects of different computational strategies on human behavior in a repeated interaction scenario. Setting up repeated interaction is the key to observing a variety of interesting behavior, since the players are able to react to strategies of their opponent. The work of Gal et al. on reciprocity [20] does evaluate computer models of human behavior in repeated interaction but does not position computer players against people.

3 Experimental Approach

Cooperative behavior is a broadly discussed and controversial subject, since there are many theories about the causes and ultimate motives of (non-)cooperative behavior [21]. We do not aim to give any final explanation of underlying motives for cooperation. Instead, we are concerned with the effect of agent behavior on human behavior. In our work, we use the word *cooperative* for interacting together willingly for a common purpose or benefit. This behavior is said to be *altruistic* if it involves a fitness cost to the proposer and confers a fitness benefit on the responder. It is said to be *egoistic* if the proposer acts only to benefit

himself by increasing his fitness. We use the notion of *helpful behavior* to refer to cooperation that does not necessarily imply a fitness cost for the proposer, but in any case yields a fitness benefit for the responder. *Reciprocal* behavior is conduct that corresponds to the behavior of others. In this study, we created a setting that encourages and instigates these different kinds of cooperation.

3.1 Colored Trails

Colored Trails (CT) is a game developed by Grosz et al. [11] which provides a testbed for investigating interactions between players whose task is to reach a goal by exchanging resources. Computational strategies and human decision making can be studied by comparing interactions in both homogeneous and heterogeneous groups of people and computer systems. CT is played on an N×M board of colored squares in which one or more squares can be designated as a goal. Each player has to reach a, possibly different, goal by exchanging chips. In order to move a player piece to an adjacent position, the player has to hand in a chip that has the same color as the square he wants to move to. It is not possible for to do diagonal moves.

3.2 Conceptual Design

Our experiment consists of two main components to examine human cooperativeness: a game and a questionnaire. In this subsection we will elaborate on the conceptual design of the game. We use the CT framework to implement an environment that is similar to an iterated Ultimatum Game (UG). In the stable outcome of an UG, each player has chosen a strategy and no player will benefit by changing his strategy under the assumption that the other players keep their strategy unchanged. This is called the *Nash equilibrium*. A Nash equilibrium would occur when the proposer offers the smallest possible amount of money (in our case: chips) to the responder and the responder accepts. It would be rational for the responder to always accept if the proposal leaves him with more than 0. However, experimental evidence shows that the proposer offers a relatively large share to his opponent and that the responder often rejects smaller positive amounts [12]. This can be interpreted as human willingness to play fair and to punish 'unfair' splits. The results of Gal et al. in [4] seem to be consistent with this scenario.

We use the UG to test whether humans show similar helpful or punishing behavior when they interact with an agent. A game consists of several rounds in which two players have alternating roles: *proposer* or *responder*. During each round, both players try to obtain all the chips they need in order to reach the goal. The proposer creates a proposal to exchange chips with his opponent. The opponent can either accept or reject this proposal. The proposer can also decide not to exchange any chips. In case of rejection, both players receive nothing. This is defined as *one-shot negotiation*. The variety of proposals that can be created show that our scenario is more complex than an UG, where the decision process in narrowed down to a pre-defined list of choices.

The CT environment enables us to create the following protocol:

1. **Orientation phase.** The player is able to get accustomed to the board and its new chips. During this phase, communication is not possible.
2. **Communication phase.** Both players try to obtain all the chips they need to reach the goal. The proposer can offer one proposal and the responder can react to it.
3. **Redistribution phase.** Agreements are enforced by the game controller (which is implemented in CT): after the proposal is accepted or rejected, the game controller will redistribute the chips (if necessary) according to the proposal.
4. **Movement phase.** If the player has the necessary chips to reach the goal, the program will execute the player's movement by moving its piece to the goal via the shortest possible path. If on the other hand the player does not have the required chips, his piece will not move at all. Again, we use an all-or-none approach to stimulate cooperation.

The scoring function is defined as follows: $s = goal + board\_size - taken\_steps$, where *goal* represents the rewarded points for reaching the goal (100), *board_size* represents the size of the board (20) and *taken_steps* is the number of steps of the taken path (variable per round). The scoring function in this experiment does not stipulate a reward dependence, but only a task dependence. Players did not receive penalties for the amount of chips they had left at the end of a round and their score was not dependent on the opponent's score in any way. Since the player scores in our scenario are independent, any act that is beneficial for the opponent does not increase the player's own fitness and can thus be considered as helpful behavior. Cooperation is stimulated by playing an iterated game. Participants have the prospect of encountering their opponent again in the game and are therefore more inclined to cooperate [22].

We use four CT variables to give an indication of the degree of altruistic or egoistic behavior of the players and their willingness to play fair:

- The offered chips. A higher amount and usefulness for the responder are considered more helpful.
- The requested chips. A lesser amount and usefulness for the responder are considered more helpful.
- The response. Accepting is considered more helpful. Rejecting might indicate an 'unfair' or non-beneficial proposal.
- The pursued path. Chips requested or accepted in order to take a suboptimal path to the goal are considered more helpful.

3.3 Agent Models

We developed three agents with different strategies: the altruistic agent (ALT), the egoistic agent (EGO), and the reciprocal agent (REC). The behavior by these agents is extreme and prototypical. Note that it is not our intention to

Egoistic Agent (EGO)	**proposer**	*offers*	randomly: a) no chips c) b) chips that are not beneficial for opponent
		requests	a) chips that it needs to reach the goal b) chips that enable shorter path
	responder	*accepts*	deals with better score than it can obtain with current chipset
		rejects	all other deals
Altruistic Agent (ALT)	**proposer**	*offers*	chips unrequired to reach goal that are useful for opponent
		requests	chips it needs for a path to goal with least costs for responder
	responder	*accepts*	a) deals in favor of itself b) deals in favor of opponent if it can reach its own goal
		rejects	deals that make it unable to reach the goal
Reciprocal Agent (REC)	**proposer**		a) behaves as egoistic proposer if favor balance =<0 b) behaves as altruistic proposer if favor balance >0
	responder		a) behaves as egoistic responder if favor balance =<0 b) behaves as altruistic responder if favor balance >0

Table 1. The different agents and their strategies

develop agents that resemble accurate human behavior or decision making. Our objective is to observe differences in human behavior that are caused by agent behavior. For that purpose, we implemented agents with different and extreme utility functions. ALT is satisfied with a suboptimal path to reach its goal and will accept almost all requests, thereby creating a good reputation for itself. EGO has a very selfish strategy: it will not grant the opponent any favors and always aims for the shortest path. REC has a mildly adaptive strategy that is a combination of the strategies of EGO and ALT. It uses a *favor balance* that keeps track of how cooperative its opponent has been in the game so far. The favor balance is calculated in the following way: *fb = 1 + pos_encounters - neg_encounters*. The default action of REC is to act altruistically, which is ensured by adding 1 to the favor balance. In case a proposal or response is judged as non-cooperative, the number of *neg_encounters* is increased, in case it is considered positive, it will increase *pos_encounters*. A response of the opponent is considered negative if it is a rejection and is judged positive if it is an acceptance. A proposal of the opponent is considered negative if it would leave the agent with a less advantageous chipset than it had beforehand, and is judged positive otherwise. Table 1 gives an overview of the three agent strategies.

We implemented these strategies to explore two hypotheses. Both hypotheses are motivated by the idea that although humans perceive agents differently from other humans, the human tendency to play fair and to encourage others to do the same will dominate.

Hypothesis 1 Cooperative behavior of the agent encourages cooperative behavior of the human opponent.
More specifically, both the altruistic and reciprocal agent, even though the altruistic agent is vulnerable to exploitation, will receive helpful behavior from their opponent.

Hypothesis 2 Egoistic (and therefore non-cooperative behavior) instigates 'punishment-behavior' of humans. We expect human players to offer the egoistic agent as little as possible and to prevent the agent from reaching its goal.

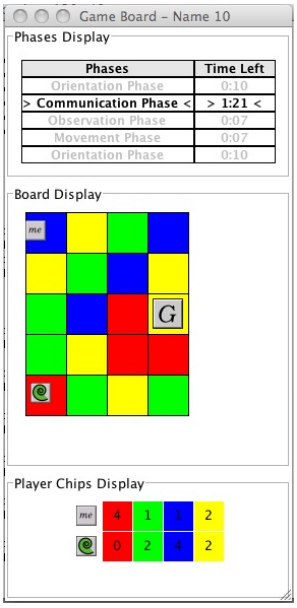


Fig. 1. The configuration of the CT game showing the board, the phases of a round and the players' chipsets

4 Experimental Design

4.1 CT Configuration

CT makes use of a wide variety of parameters that allow for increasing complexity of the game along different dimensions. The games were played with *full board visibility* and *full chip visibility*: both players had complete knowledge of the colors of the squares, both their positions on the board and the chip distribution. The full board and chip visibility allowed players to make deliberate decisions about helping their opponent or not. The PathFinder is an additional tool for players that shows the chips needed to take the shortest path to the goal. In this experiment, players were only allowed to see their own chips in the PathFinder and were not shown any paths longer than the shortest one.

Figure 1 shows our basic CT configuration. The game is played on a 4×5 board with one square designated as a goal. The scoring function, board configuration and positions of the players and the goal on the board remain the same throughout the game. The player to start and the role he[1] performs (proposer or responder) are determined randomly. In each round, the game manager randomly allocates one of ten different sets of chips to the players. The chipsets

[1] Both male and female subjects as well as gender-neutral agents were involved in our experiment. For purposes of convenience and clarity we will refer here to all subjects as male. This is however completely arbitrary.

are composed to stimulate 'interesting' negotiation behavior. There are always enough chips in the game for both players to reach the goal, but the players never both start with a chipset that allows reaching the goal via the shortest path. The strategies of the agents are reflected in the chips they are willing to transfer. For example, it is considered altruistic to propose a chip distribution that allows the responder to reach the goal but leaves the proposer with a longer path to the goal. 'Helpful behavior' entails a broader range of conduct in which a proposer can perform actions that increase the fitness of the responder without necessarily decreasing the proposer's own fitness. In this scenario, a proposer displays helpful behavior if he transfers chips to a responder to help him reach the goal while the proposer can still reach the goal via the shortest path.

4.2 Experimental Setup

A total of 30 subjects participated in the experiment. All participants were graduate and undergraduate students between 20-30 years of age. 80% of them studied Artificial Intelligence, 20% were enrolled in other studies. Almost 50% of the participants were female. Each subject participated in 4 games, adding up to a total of 120 games played. Participants were instructed to perform a non-competitive task: they were to try to maximize their own scores, not to minimize other players' scores. This is also reflected in the scoring function, since the majority of the points can be received by reaching the goal, not by choosing the shortest path. It follows that there is no strictly competitive or zero-sum iteration. A small financial bonus payment would be awarded to the player who obtained the highest score. Players were aware of this, but since the bonus would be awarded to all player with the highest score, we expect the bonus did not stimulate players to minimize their opponent's score, but only to maximize their own score.

Given that our main goal was to examine the differences in human behavior when confronted with (non-)cooperative behavior of an agent, we had to compare this behavior in some way to the behavior of humans towards a human opponent. For this reason, participants played three games against agent opponents and one game against another human participant. The group that played against human opponents was used as a control group to demonstrate the degree of cooperativeness of the players. We expected that telling the participants they would be playing against computer agents could alter their behavior. In order to investigate the effect of knowledge and beliefs of their opponent on the acceptance level of 'unfair' or non-cooperative behavior, we deliberately manipulated their knowledge and beliefs.

Hence the general setup included four independent variables:

1. **Nature** of the player: human or agent.
2. **Strategy** of the agent: altruist, egoist or reciprocal.
3. **Knowledge** of the nature of opponent: True Belief (TB), False Belief (FB), No Belief (NB).
4. **Order** of played opponents: egoist, altruist, reciprocal, human.

We created three groups of ten participants each and combined them with different belief conditions. The belief condition provided the participants only with information about the nature of the opponent; the strategy of the agents was never disclosed. The game controller randomly determined the order of the opponents for each of the three groups. Each game consisted of ten rounds. In the TB-condition we told the participants the truth about whether they were playing against an agent or against another human. In the FB-condition we misled them by announcing their opponent would be a human when in fact it was an agent, and the other way around. Unlike in the other conditions, we did not give the players any information at all about their opponent in the NB-condition.

4.3 Evaluation

At the end of each game, participants were asked to fill out a questionnaire. This questionnaire provided us with insights of how different aspects of cooperation come about. They answered questions about the nature and strategy of their opponent and about their own strategy and cooperativeness. We used this information to find out how the participants perceived their opponents and how this influenced their cooperative behavior.

The logs of each CT game contained all the vital information of the game, such as score, the proposals and responses made, whether both players reached their goal and if so, how many steps it took. These logs provided us with information of how often players made fair trades or helped their opponents.

5 Results

We hypothesized humans to behave more cooperatively towards opponents that behave cooperatively or altruistically themselves (H1). We also expected that egoistic behavior would be punished (H2). Our results support both hypotheses. However, more extensive research with a larger number of participants has to be done to significantly demonstrate these results.

The questionnaire shows that our agents' strategies correctly represent the different types of behavior. 100% of the participants found ALT moderately to very cooperative: 83% found it very cooperative and 17% found it partially cooperative. Interestingly, REC was also considered very cooperative: 97% of the subjects perceived it as cooperative. This can be explained by the fact that the first action REC takes is an altruistic one; that is, it only adopts an egoistic strategy when the opponent treats it in an egoistic way. The majority of the players found EGO to be not cooperative at all but surprisingly a considerable number of subjects believed the agent to be partially (30%), or occasionally even fully (7%), cooperative. In comparison, 53% of the subjects considered their human opponents moderately to very cooperative. As it turned out, participants judged the egoistic agent as being cooperative because it sometimes offered chips when it did not need anything in return. It did not seem to matter that these chips were of no use to the responder.

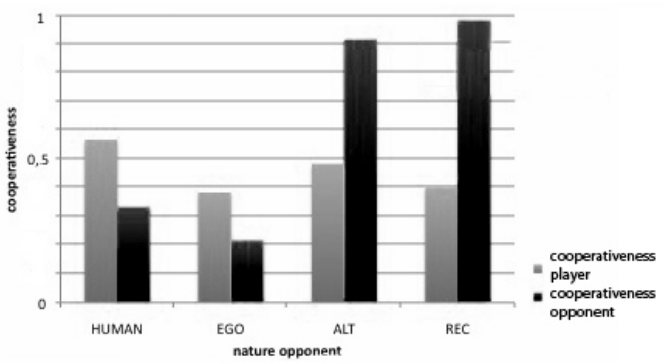


Fig. 2. Results of how cooperative humans perceived themselves and their opponents to be. Scale 0-1: 0 = non cooperative, 0.5 = partially cooperative, 1 = very cooperative

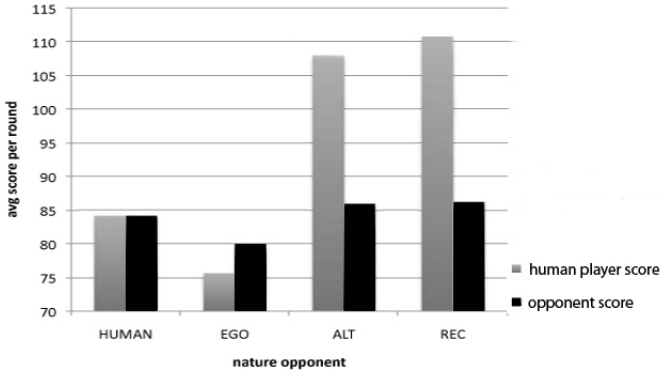


Fig. 3. Average score per round

The data from our survey shows that human participants regard themselves more cooperative towards opponents who they perceive to be cooperative. These results can be found in Fig. 2. The collaboration of human players increases as their opponent shows more cooperative or altruistic behavior. This is most clear when we compare the degree of cooperativeness towards altruistic and egoistic agents, respectively 0.49 and 0.39 on our scale from 0 to 1. Furthermore, players consistently identified themselves as more cooperative than their human and egoistic opponents.

Figure 3 shows the average score of subjects per round, playing against different opponents. The categories on the horizontal axis represent the experimental condition, i.e., the category 'ALT' refers to the experimental condition of the participants playing against an altruistic opponent. The categories show the av-

Table 2. Average score of the human players and their opponents in the different belief conditions

	Player Score	Opponent Score			
		HUM	EGO	ALT	REC
TB	90.9	76.1	74.7	85.4	84.5
FB	95.3	84.4	79.8	88.7	84.3
NB	97.7	92.0	85.4	83.6	89.8

Table 3. The percentage of human players that correctly identified the nature of their opponent.

Nature opponent	% players that identified opponent's nature
HUMAN	63 %
EGO	50 %
ALT	83 %
REC	83 %

erage of all belief conditions (True Belief, False Belief and No Belief). The two main results are the following:

1. The score of the agent increases if it has a more cooperative strategy.
2. Human cooperation with the egoistic agent does pay off, because the average score of both the human player and the agent exceed the minimum score average of 70.

Chipsets were thus distributed that in some rounds, one of the players was able to reach the goal with the initial distribution. Without any cooperation, players were able to obtain an average of 70 points per round. In our experiment, the altruistic and reciprocal agent have average scores of 85,9 and 86,2 respectively. The egoistic agent stays behind with an average score of 80. This also demonstrates that the perceived increase in human cooperativeness is actually reflected in the game.

Interestingly, humans do not fully exploit altruistic players. The average score per round of both ALT and REC are higher than the ones of EGO and the human opponent in the human-human condition. Altruistic agents reach their goal slightly more often (75% of the time) than both human (73% of the time) and egoistic players (69% of the time). The questionnaires reveal that players were prepared to give the altruist chips that would help it reach its goal. The logs confirm this: players transferred on average 4.0 chips per game to EGO and 5.6 chips to ALT.

In the different game conditions players received no, false or true information about the nature of their opponents. Table 2 shows the average score of the players in these conditions. The 'Player Score' denotes the average score of the human players on all rounds. The 'Opponent Score' column shows the average scores of the different types of opponents. Note that the HUM and EGO scores increase as the amount of (true) information about the nature of their opponent decreases. Since human opponents were perceived as rather uncooperative (cf. Fig. 2), this might indicate that egoistic strategies perform better as they have to deal with more uncertainty.

At the end of the game, players were asked to identify their opponent as human or agent. The results can be found in Table 3. Remarkably, EGO was correctly identified only at chance level, i.e., in 50% of the cases the egoistic

agent was mistaken for human. A possible explanation might be that people expect other people to behave selfishly and egoistically in negotiations and as a consequence this behavior is perceived as more human-like [23]. The results of the questionnaire clearly show that the behavior of the altruistic agent is seen as 'stupid', 'dumb' and 'too cooperative to be human'. This corresponds to findings of Kraus and Grosz in [11], who suggest that system designers should build cooperative agents because people expect agents to be cooperative.

6 Discussion

Our experiment provides preliminary insights in negotiation behavior between humans and agents with contradictory interests. Our results suggest that cooperative agents stimulate cooperative human behavior and that humans have certain expectations of the behavior of human and agent opponents. These insights can be applied in more complex and realistic domains, such as e-commerce and auctions. However, the applicability of these finding is limited and is restricted to negotiation settings within a controlled environment. Less regulated interactions, e.g., in social networks, may rely on alternative expectations that require more dynamic, adaptive and complex behavior. We intend to take a step in this direction in future work and use small social networks (teams) to examine cooperative behavior between team players with partly contradictory goals. We also plan to extend our agent strategies to more complex and adaptive ones so as to make them more realistic.

Taking all this into account, our results provide insights into social aspects of human-agent negotiation in specific domains, which can be used to further explore social relations between humans and agents.

7 Conclusion

In this paper we have explored the question of how the behavior of a software agent affects the cooperativeness of its human opponent in a negotiation setting. Initial results show that humans behave more cooperatively towards agents that negotiate with them in a cooperative way. We also find that humans tend to punish egoistic and unfair play by behaving non-cooperatively themselves. Furthermore, egoistic agents are more likely to be identified as humans than altruistic agents.

Our results are not surprising in the wider context of recent work on human behavior analysis and game theory [10, 12]. They confirm the expected results that it is beneficial to act cooperatively in order to induce cooperativeness. However, they are important since previous work has yielded contradictory results which makes it difficult to make assumptions about social conducts between humans and agents. Although our experiment is small in scale, our results provide a foundation for further research in this area.

In order to achieve statistical significance of the results, in the future, we will extend our experiment to a larger number of participants. We plan to extend the agents' behaviors to more complex and adaptive ones.

Acknowledgements. We thank Ya'akov Gal and Bart Kamphorst for helpful comments and assistance with the Colored Trails system. This research is funded by the Netherlands Organization for Scientific Research (NWO), through Veni-grant 639.021.509.

References

1. Jennings, N., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous agents and Multi-Agent Systems (1) (1998) 257–306
2. Riedl, M., Saretto, C., Young, R.: Managing interaction between users and agents in a multi-agent storytelling environment. AAMAS (2003)
3. Rajarshi, D., Hanson, J., Kephart, J., Tesauro, G.: Agent-human interactions in the continuous double auction. IJCAI (2001)
4. Gal, Y., Pfeffer, A., Marzo, F., Grosz, B.: Learning social preferences in games. AAAI (2004)
5. Sanfey, A., Rilling, J., Aronson, J., Nystrom, L., Cohen, J.: The neural basis of economic decision-making in the ultimatum game. Science (300) (2003) 1755–1758
6. Sierhuis, M., Bradshaw, J., Acquisti, A., van Hoof, R., Jeffers, R., Uszok, A.: Human-agent teamwork and adjustable autonomy in practice. i-SAIRAS (2003)
7. van Wissen, A., van Diggelen, J., Dignum, V.: The effects of cooperative agent behavior on human cooperativeness (short paper). AAMAS (2009 (to appear))
8. Camerer, C.: Behavioral Game Theory: Experiments in Strategic Interaction. Princeton University Press (2003)
9. Axelrod, R.: The Evolution of Cooperation. Basic Books (1984)
10. Blount, S.: When social outcomes aren't fair. Organizational Behavior and Human Decision Processes **63**(2) (1995) 131–144
11. Grosz, B., Kraus, S., Talman, S., Stossel, B., Havlin, M.: The influence of social dependencies on decision-making: Initial investigations with a new game. AAMAS (2004)
12. Guth, W., Schmittberger, R., Schwarz, B.: An experimental analysis of ultimatum bargaining. Journal of Economic Behavior and Organization (3) (1982) 367–388
13. Loewenstein, G., Thompson, L., Bazerman, M.: Social utility and decision making in interpersonal contexts. Journal of Personality and Social Psychology (1989)
14. Ross, M., Fletcher, G.: Attribution and social perception. In Lindzey, G., Aronson, E., eds.: Handbook of Social Psychology. Random House (1985)
15. Kagel, J., Roth, A.: The Handbook of Experimental Economics. Princeton University Press (1995)
16. Weiss, G., Sen, S.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
17. de Jong, S., Tuyls, K., Verbeeck, K.: Fairness in multi-agent systems. The Knowledge Engineering Review (2008)
18. Danielson, P.: Competition among cooperators: Altruism and reciprocity. In: PNAS. (May 2002)
19. Hogg, L., Jennings, N.: Socially intelligent reasoning for autonomous agents. IEEE Trans on Systems, Man and Cybernatics - Part A (2001) 381–399
20. Gal, Y., Pfeffer, A.: Modeling reciprocity in human bilateral negotiation. AAAI-07 (2007)
21. Katz, L., ed.: Evolutionary Origins of Morality: Cross-Disciplinary Perspectives. Imprint Academic. (2000)
22. Binmore, K.: Fun and Games: a Text on Game Theory. D.C. Heath and Company (1992)
23. Epley, N., Caruso, E.M., Bazerman, M.H.: When Perspective Taking Increases Taking: Reactive Egoism in Social Interaction. SSRN eLibrary (2005)
24. Nassiri-Mofakham, F., Ghasem-Aghaee, N., Nematbakhsh, M., Baraani-Dastjerdi, A.: A personality-based simulation of bargaining in e-commerce. Simulation Gaming **39**(1) (2008)