Neglect Tolerant Teaming: Issues and Dilemmas[*]

Michael A. Goodrich, Jacob W. Crandall, and Jeffrey L. Stimpson
Computer Science Department
Brigham Young University
Provo, Utah, USA 84602

Abstract

In this paper, a brief overview of neglect-tolerant human-robot interaction is presented. Recent results of a neglect-tolerance study are then summarized. The problem is then posed of how neglect tolerance affects how a human interacts with multiple robots, and a scenario is constructed that illustrates how multiple robot management can produce a problem with the form of a prisoner's dilemma. An abstraction of this prisoner's dilemma problem is then presented, and two robot learning algorithms are outlined that may address key points in this abstracted dilemma.

Introduction

In the phrase *human-robot interaction*, the term *interaction* consists of a robot *autonomy level* as well as an *interface* element. This interaction is illustrated in Figure 1 for a situation where a human interacts with a remote robot over a communication link. The human gathers information about the world (and the robot's status) through messages sent from the robot to the human, and then manages the robot by passing messages to it. This loop is governed by the interface that facilitates information dissemination and robot instruction. The robot then modulates the instructions from the human to generate actuator signals in some manner which takes into consideration what the robot senses. This modulation is governed by the autonomy algorithms present on the robot.

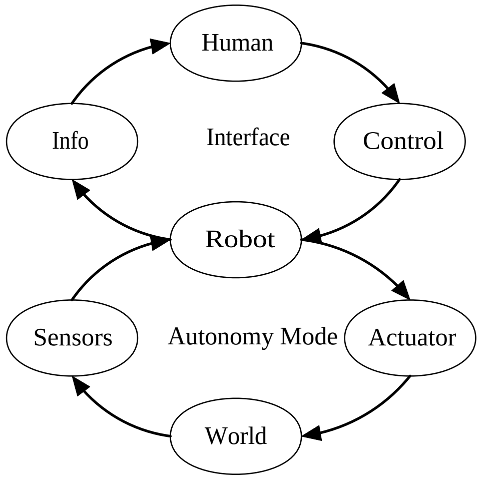

Figure 1: Neglect tolerance.

Much has been written about adjusting robot autonomy levels, but dynamic autonomy requires some method for choosing the right autonomy for the circumstances. In this paper, we identify the variables that determine efficient interaction using the concept of *neglect-tolerance*. We then address how teams of semi-autonomous, semi-independent robots could learn to coordinate in a way that utilizes human attention efficiently, and identify some issues and dilemmas that can arise using the set of autonomy levels discussed herein. Finally, we speculate on two possible algorithms for addressing these dilemmas.

Neglect-Tolerant Interaction

Concept

We can summarize the concept of neglect tolerant interaction[1] as follows (Crandall & Goodrich 2002; Goodrich *et al.* 2001): *Performance of a semi-autonomous robot declines as human attention is spent on other tasks and/or as the complexity of the world increases.* This relationship between robot performance, neglect time, and complexity is conceptualized in Figure 2.

The precise shape of the neglect-tolerance curve for a particular autonomy mode is dictated by (a) *neglect impact*, the decline in performance that occurs when human attention is dedicated to a secondary task, and (b) *interface efficiency*, the way an interface assists a human to regain robot situation-awareness and re-direct a robot once attention is dedicated to the robot. Combining these two factors and setting a level of required performance determines how much time can be spent on tasks other than managing a particular robot. This combination is illustrated in Figure 3. In the figure, the robot begins at a standstill and the human instructs the robot (task 1) until performance achieves a peak level.

[*]This work was supported in part by DARPA/DOI contract #NBCH1020013

[1]The reader is referred to (Sheridan 1992; Wickens & Hollands 2000) for a survery of related concepts.

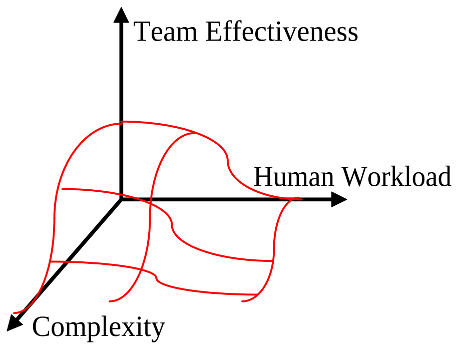

Figure 2: Neglect tolerance.

The human then turns attention to a second task (task 2) during which time performance deteriorates while the robot is neglected. Sometime before this level of performance declines below an acceptable level, the human turns attention back to the robot (task 1) and instructs the robot enough for performance to increase to peak performance again.

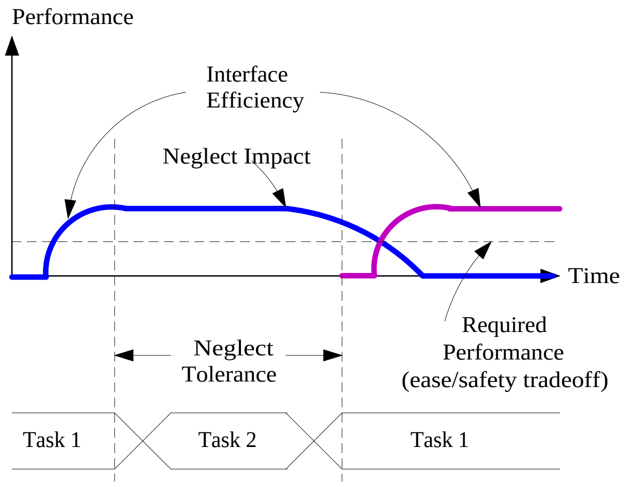

Figure 3: Neglect tolerance is determined by neglect-impact and interface efficiency. These conceptual plots are shown for a fixed level of complexity.

For a team of semi-autonomous, semi-independent robots, the neglect tolerance characteristics of each robot constrain how attention needs to be shared between robots. For example, time spent servicing robot A is time spent neglecting robot B, and this means that the number of robots that can be managed by a single human is limited by the neglect tolerance of the robots. Thus, to efficiently manage multiple robots, it is necessary for the robots to coordinate in such a way that human attention is used well.

Multiple Autonomy Modes We have developed a suite of robot autonomy levels and verified that these levels work on real robots. To explore how these autonomy levels might be affected by neglect, we set up a series of simulator experiments with 32 undergraduate students drawn from a class of computer science and electrical engineering students. The three autonomy modes are: *teleoperation*, *point-to-point*, and *scripted*.

The *teleoperation* autonomy mode uses the shared control algorithm described in (Crandall & Goodrich 2001). In this control scheme, the human guides the robot via a joystick while watching a video stream from the robot's camera. The robot modulates the direction vector given by the joystick by accounting for obstacles sensed by the sonars. This balance allows the robot safely to go in the general direction requested by the user.

The *point-to-point* autonomy mode also uses the shared control algorithm. In this control scheme, the human operator clicks the mouse on various locations within the video stream window to tell the robot what to do at the next branching point that it finds (e.g., click on the right side of the video stream if the robot is to turn right at the next intersection). The robot can be told to turn right, turn left, or go straight through the intersection. In addition, the human can direct the robot to go backwards, spin right, or spin left. If no input is given, the robot defaults to going straight. The interface confirms the human input by superimposing a directional arrow on the video stream.

This interaction scheme is more autonomous than the teleoperation system because the human must only provide an input vector for every branching point (i.e., intersection) that the robot faces. Additionally, if the robot gets turned around by obstacles, misses an intersection, or thinks it has found an intersection when it, in reality, has not, the operator must provide additional help. In this way the robot can move through somewhat cluttered environments effectively.

The *scripted* autonomy mode also uses the shared control algorithm. The interface for this scheme is a god's-eye map of the world where the human clicks to establish waypoints. The human must not only put a waypoint at every intersection (i.e., the robot does not do any path planning) but also use waypoints to guide the robot through some complex obstacles. A script of these waypoints is created and then executed by the robot. The robot derives a direction vector from the nearest waypoint, uses the shared control algorithm to travel to the waypoint, and then reorients to the next waypoint. The interface confirms that a waypoint has been created (or destroyed) by placing (or removing) a colored block on the map.

This autonomy mode is more autonomous than point-to-point because the robot can navigate an entire route from start to the goal without intervention (if the waypoints are appropriately placed). If no waypoint is found, the robot stops and waits.

Experiment and Results In the experiments, 32 undergraduate computer science and electrical engineering students were subjects. Each subject was asked to control two simulated[2] robots from their current location to a goal. The

[2]Although simulated robots are not sufficient to validate an interface or a suite of robot algorithms, they are sufficient to allow neglect tolerance to be characterized. In the context of this paper, these characterizations are sufficient to identify the resulting robot dilemmas.

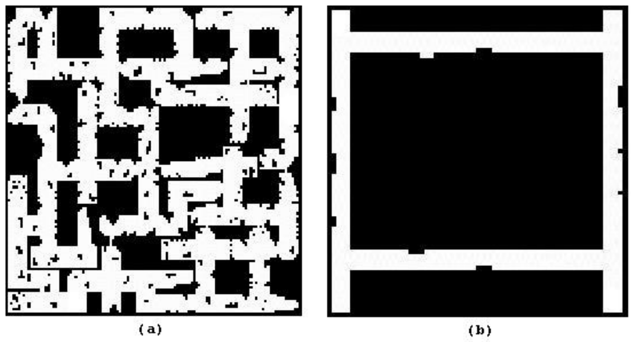

Figure 4: An extremely simple world (right) and an extremely complex world (left).

goal was displayed on a god's eye map, and the subject was instructed to guide a simulated robot through a simulated world to the goal. When the robot reaches the goal, a new goal is placed randomly in the world. The information element included a god's eye view of the world as well as the position of the robot in that world. Additionally, the 16 sonar values, compass readings, and video stream was displayed.

Several different worlds of varying complexity were used. Typical worlds include environments with low clutter and low branching, low clutter and high branching, high clutter and low branching, and high clutter and high branching so as to model many kinds of environmental complexity. Two worlds of extreme complexity are shown in Figure 4.

Complexity was calculated using four measurements obtained from the robot. Three of these measurements indicate the amount of clutter in the environment. First, sonar values will tend to change more rapidly in cluttered environments than uncluttered environments. Second, the autonomous algorithms on a robot will tend to cause it to change directions (encoded as steering entropy (Boer *et al.* 2001)) as it moves around obstacles. Third, a robot's velocity will tend to fluctuate as it moves around obstacles (clutter). Measurements are taken from these three observations, and are averaged together to estimate the clutter C_c of the environment (a value between 0 and 1). Branching complexity is predicted by observing the average number of afforded directions a robot can take over a period of time. This value is also normalized between 0 and 1 (0 for no branching whatsoever, and 1 for a high number of branches) to predict the branching factor C_b of the current environment. The complexity of the environment is then estimated by

$$C = w_c C_c + w_b C_b.$$

Note that the complexity estimates need not be perfect as long as the relative contributions of various elements are balanced. The reason for this is that complexity can be defined as those environmental characteristics that cause performance to decline, and the measurements obtained herein demonstrate the the complexity estimate is sufficient for this purpose.

For point-to-point and scripted modes, the operator was given as much time as was needed to service the robot[3]; this means that once the operator started to instruct the robot, they controlled it for as much time as "felt natural" to them [from the instructions given to the subjects]. When the operator was done servicing the robot, he/she clicked a button, and the robot was neglected for a pre-specified interval of time (up to 40 seconds). During this time, the operator either (a) serviced the second robot[4] (if it had been neglected for the amount of time specified by the experiment design) or (b) performed a secondary math task. The secondary math task required that the human select an answer to a two-digit arithmetic (addition or subtraction) problem from a list of four possible answers. The math problem was presented in the same position as the robot video stream, and the answer was selected by clicking on the appropriate answer. Servicing the second robot and doing the secondary math problems required both cognitive effort, attention management, and motor response generation. This means that working memory, motor responses (Wickens & Hollands 2000), and attention (Pashler 1997) were loaded.

Neglect Tolerance Cureves After servicing the second robot or performing the secondary math task, the video stream for the first robot again appeared and the operator again serviced the robot. Data was averaged across subjects. The neglect tolerance curves for point-to-point and scripted autonomy modes are shown in Figures 5-6.

The neglect-tolerance random process for the (shared control) teleoperation mode is not shown because it is uninteresting. Performance goes to zero for any neglect interval longer than about 0.5 seconds. This occurs because subjects take their hands off the joystick to use the mouse, and when the joystick is not controlled the robot stops. As complexity decreases, there is a decrease in the robot's performance, as expected.

Figure 5 displays the mean of the neglect-tolerance random process for the point-to-point mode. Although the data is noisy, the general trends match the hypothesized shape of Figure 2. The plot of the expected (mean) performance of the robot shows that as the robot is neglected, performance decreases over time. Additionally, as the environment becomes more complex, performance decreases as well. In very complex environments, the robot performs quite poorly in this interaction scheme.

Figure 6 displays the mean of the neglect-tolerance random process for the scripted mode. Although the data is noisy, the general trends again match the hypothesized shape of Figure 2. In very complex environments, the robot performs quite poorly in this interaction scheme. Depending on the workload requirements of the system, this interaction scheme may not be appropriate for some complex

[3]Because the robot was serviced for a sufficient period of time, the neglect random process is not necessarily an accurate estimate for instances in which a robot is not serviced sufficiently.

[4]Robots were identified with a color-coded icon in the god's eye view, and the display background changed colors to indicate which robot was being serviced. This helped reduce mode confusion.

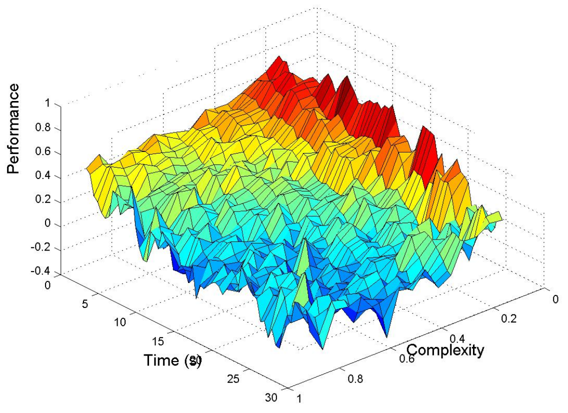

Figure 5: A graph of the mean of the neglect tolerance random process for the point-to-point autonomy mode.

worlds/environments unless the human dedicates a lot of attention to it.

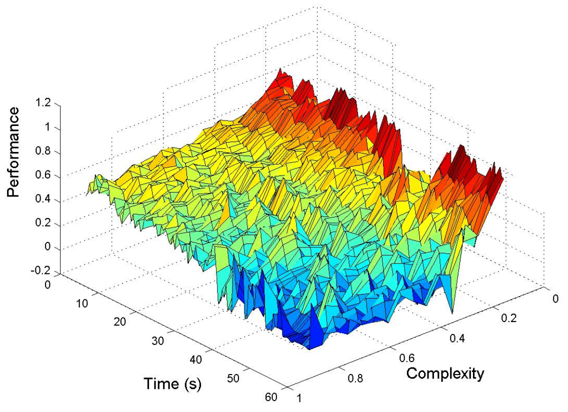

Figure 6: A graph of the mean of the neglect tolerance random process for the scripted autonomy mode.

Note that, in the interest of space, we have not presented results from variability. Variance in the random processes increases when complexity increases indicating that complex worlds affect how consistently a robot performs.

Interface Efficiency Curves Figure 7 illustrates the interface efficiency for the scripted autonomy mode. As predicted, more time-on-task results in greater instantaneous performance of the robot, but it takes some time for peak performance to be obtained. More complex worlds reduce the efficiency of the interface. Figures 8 and 9 present the same information for the point-to-point and teleoperated autonomy modes.

These three plots display some anomalies in how complexity impacts performance, and may indicate that some

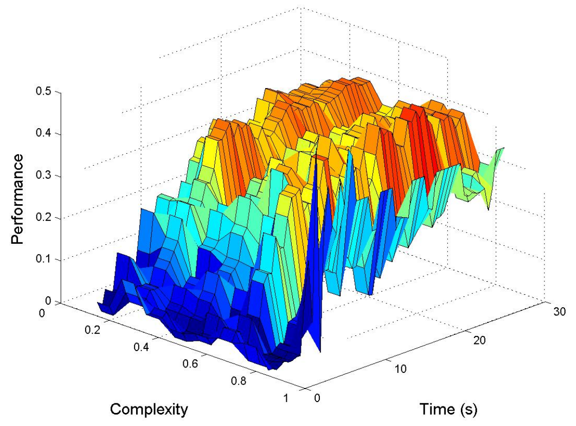

Figure 7: A graph of the mean of the interface efficiency random process for the scripted autonomy mode.

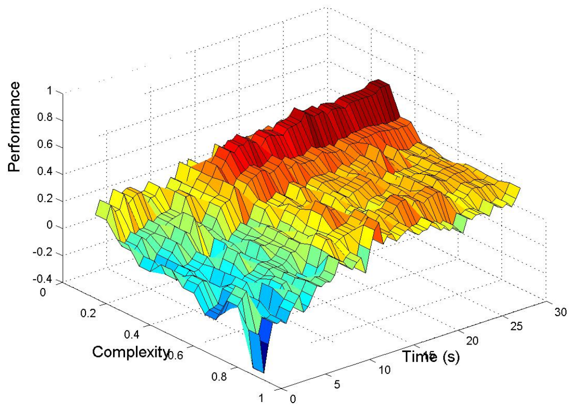

Figure 8: A graph of the mean of the interface efficiency random process for the point-to-point autonomy mode.

adjustments to the complexity estimate need to be made. There are also some strange things that happen along the performance axis, such as the slight decrease in teleoperation efficiency after an initial improvement, that may lead to further insight into how humans interact with robots.

When comparing the results, it is important to note that the scales are different along both the performance axis and the time axis. Thus, teleoperation has a higher possible performance level for nearly all world complexities. Furthermore, teleoperation reaches its maximum level quicker than the scripted and point-to-point modes.

Sharing Attention

The Game: A Motivating Example

In this section, we want to present a simple illustration of how neglect tolerance can create a social dilemma if a human is trying to manage multiple robots. Consider the prob-

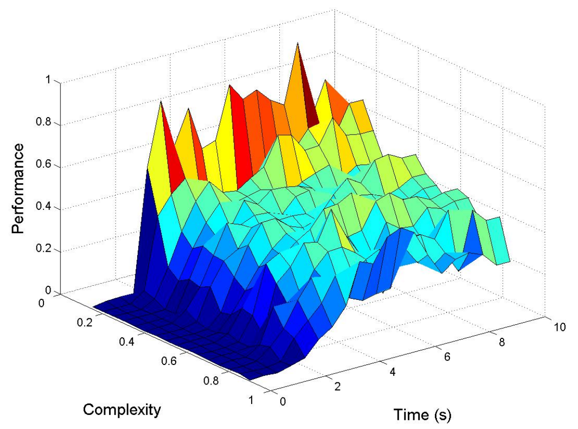

Figure 9: A graph of the mean of the interface efficiency random process for the teleoperation autonomy mode.

lem of a human manager interacting with two robots. Each robot has adjustable autonomy levels, and the human and robots must manage human intention in such a way as to maximize team performance. In this scenario, suppose that the team must explore an environment as quickly as possible. The robots must learn the best way to interact with the human; such interaction includes selecting an appropriate level of autonomy and organizing relevant information in such a way to allow the human to interact.

Suppose that the robots can select from the three levels of autonomy described herein: teleoperation, point-to-point, and fully autonomous. Associated with each of these autonomy levels is an associated neglect curve. The robots must operate in the world diagrammed in Figure 10. In the

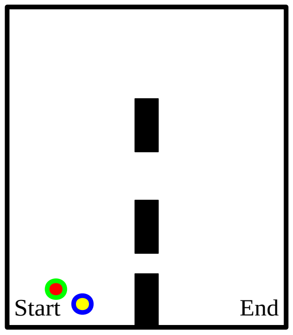

Figure 10: *A navigation world where a dilemma might arise.*

world, the two robots must navigate through one of three corridors. The southern corridor is too narrow (e.g., it has high complexity) for the robot to autonomously navigate; a robot passing through this corridor requires teleoperation. The middle corridor is wider than the southern and narrower than the northern (e.g., it has moderate complexity); one or two robots can navigate the corridor with waypoint-driven assistance from the human. The northern corridor is wide (e.g., it has low complexity) and can easily be navigated by one or two autonomous robots.

Suppose that the two robots begin on the same side of the room and must pass through one of the corridors to reach their respective goal locations. Each robot must make choices among the following options:

- Go to the southern corridor, switch to teleoperation mode, and request the manager to teleoperate them through the corridor.
- Go to the middle corridor, switch to waypoint-driven mode, and request the manager to guide them through the corridor.
- Go to the top corridor and autonomously drive through the corridor.

There are a couple of ways that we could solve this problem. First, we could have the human choose an autonomy level for each robot, but this just means that the human has another task to do (the autonomy level selection task) which, in turn, means that robots will be neglected longer. Second, we could define a social choice mechanism and negotiation protocol wherein robots follow scripted interactions to determine which robot gets which autonomy mode. Third, each robot could (autonomously) make a decision about what autonomy level to choose and they could learn which states translate into which autonomy modes. It is this latter problem with which this paper is concerned.

	D	C
D	(P,P)	(T,S)
C	(S,T)	(R,R)

Table 1: Payoffs for a traditional Prisoner's Dilemma game (Axelrod 1984). For a dilemma to arise, we must have T>R>P>S and $\frac{T+S}{2}$ >R. T encodes the "temptation" payoff, S encodes the "sucker's" payoff, R encodes the mutual "reward" for cooperating, and P encodes the "penalty" for mutual defection.

Such a scenario is a type of social dilemma, similar to the prisoner's dilemma shown in Table 1. When both robots defect (chooses teleoperation), a time of P units elapses (from the confusion of the operator plus the time to navigate through the middle corridor). When one robot defects (choose teleoperation), the other robot cannot choose teleoperation nor point-to-point control since all operator attention is consumed by the first robot; the second (non-teleoperated) robot must act autonomously and receives the S payoff (because it takes S units of time to reach its goal) while the first (teleoperated) robot receives the T payoff (because it takes only $T < S$ units of time for it to reach its goal). When both robots choose to cooperate (i.e., they choose the point-to-point option), both receive the R payoff (because it takes R units of time, $T < R < S$, for each of these robots to reach their goals).

The Game: A First-Level Abstraction

Such dilemmas can take many different forms, so it is useful to find a type of abstraction that can be used to test different algorithms. In this section, we summarize an abstract world that has many of the important elements of these dilemmas. Consider the following world. In this world, two agents

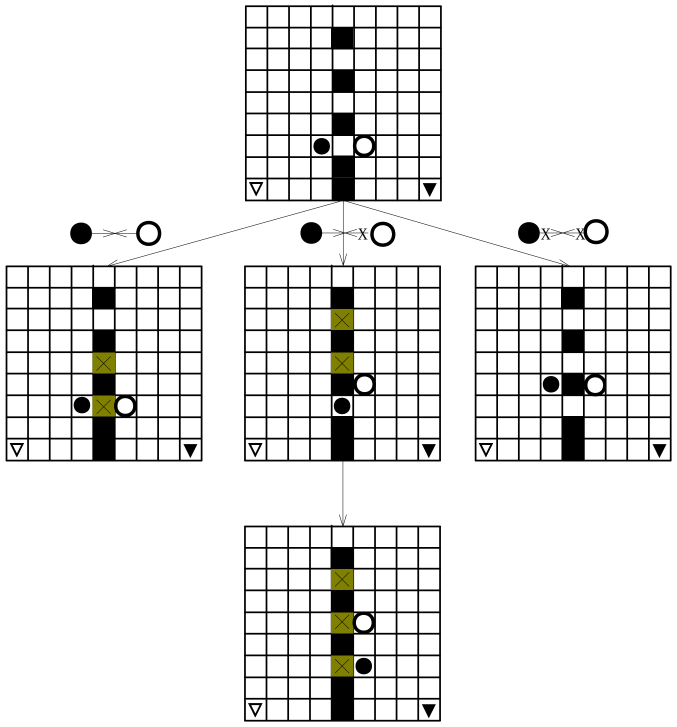

Figure 11: A prisoner's dilemma navigation world.

CIRCLE and DOT, are placed on opposite sides of a partition in mirror image locations. CIRCLE's goal (represented by a triangle) is placed in the lower left corner of DOT's territory, and DOT's goal is placed in the lower right corner of CIRCLE's territory. Each agent desires to reach its corresponding goal in minimum time.

In this world, agents are allowed to simultaneously occupy a cell. However, if both agents simultaneously attempt to enter cell (5,3), the southernmost passage (gate 1) through the partition, then a door closes and neither agent is allowed to enter. In addition to gate 1 being closed, gate 2 (in state (5,5)) is also closed; this is depicted on the left-center grid in Figure 11. If only one agent tries to pass through gate 1 then it is allowed to pass but gates 1, 2, and 3 (in state (5,7)) close forcing the other agent to travel through the northernmost passage through the partition to reach its goal; this is depicted on the center two grids in the figure. Finally, if neither agent tries to pass through gate 1 then all doors stay open all the time; this is depicted in the right-center grid in the figure.

Now, consider when both agents are adjacent to gate 1 in the world, as depicted in the figure. Each agent has two qualitatively different choices: choice D is to try to go through the gate, and choice C is to move elsewhere. Depending on the joint choices made by the agents, the length of the minimum-length path for both agents is changed. In game matrix form, the minimum number of actions required to reach the goal state are, assuming optimal choices thereafter (except in the right-center grid where we assume that the agents pursue the path through gate 2) shown in Table 2. When this table is compared to the traditional pris-

CIRCLE/DOT	D	C
D	(16,16)	(7,19)
C	(19,7)	(11,11)

Table 2: Minimum number of actions given a particular strategy. Note that the minimum path through gate 3 to the goal is 15 steps; we use the number 16 as the effective path length because we consider the case when agents first attempt to go through gate 1, thus increasing the minimum number of required actions..

oner's dilemma formulation, shown in Table 1, the prisoner's dilemma becomes apparent.

Adapting to Human Limitations

Given that social dilemmas can arise when multiple robots try and interact with a human, we propose having the robots learn which autonomy levels are appropriate for given world complexities and human attentional demands. Current algorithms in multi-agent learning do not seem appropriate for these dilemmas because they emphasize either learning a Nash equilibrium or they require a centralized learner to coordinate the robots. A Nash equilibrium is inappropriate because the equilibrium in the Prisoner's dilemma is worse, from the human manager's perspective, than mutual cooperation. A centralized learner is inappropriate because one purpose of having multiple robots is to take advantage of the distributed computing, the robot mobility, and the robustness to failure that multiple robots provide. Thus, we want to identify learning algorithms that allow robots to succeed in social dilemmas without a centralized arbiter and still be robust to robot failures. Our research in this area is still in progress, but we present overviews of two possible algorithms: Relaxed Negotiation and BEANS.

Relaxed Negotiation

The relaxed negotiation algorithm is a variant of Karandikar et. al 's satisficing algorithm (Karandikar *et al.* 1998). The idea of their algorithm is that, in the repeated play of a prisoner's dilemma game, an agent can update their aspirations and change their behavior if such a behavior does not meet the aspiration. Formally, suppose that a is the set of actions available to the agent. In the simple prisoner's dilemma, the actions are cooperate, C, and defect, D. After each agent makes its choice, payoffs are revealed according to the payoff matrix. Let $u_i(a_1, a_2)$ denote the payoff to agent i when agent 1 plays a_1 and agent 2 plays a_2. An aspiration level α_i represents the satisficing aspiration level of agent i (Simon 1996). At the next time step, the decision algorithm in Table 3 is used to make a choice. In words, the algorithm says to stay with what you did last time if the payoff exceeded the aspiration; otherwise switch. After each reward is received,

Choice at time t	Choice at time $t-1$	Condition
C	C	$u_i(a_1, a_2) \geq \alpha_i$
D	D	$u_i(a_1, a_2) \geq \alpha_i$
C	D	$u_i(a_1, a_2) < \alpha_i$
D	C	$u_i(a_1, a_2) < \alpha_i$

Table 3: Karandikar's satisficing choice algorithm.

aspirations are updated according to

$$\alpha_i \leftarrow \lambda\alpha_i + (1-\lambda)u_i(a_1, a_2).$$

Karandikar et. al showed that if this update rule was perturbed by a small amount of noise then the two agents nearly always play mutual cooperation.

Stimpson (Stimpson 2002) modified this algorithm by eliminating the perturbation in the aspiration level and instead requiring that aspiration levels start high. He also let any choice be made with equal probability if the aspiration level was not met. Under such circumstances, a high value of λ (which translates into slowly decreasing the aspiration level), implies that mutual cooperation emerges with guaranteed high probability. Thus, Stimpson turned the satisficing update into a relaxation search where the aspiration levels are slowly relaxed from a high level to a lower level. This relaxation allowed, in essence, agents to bargain from an initially high level of aspiration until they found a solution where both agents were satisfied. Neither agent has an incentive to defect from this solution (it is a satisficing equilibrium in the sense of (Stirling, Goodrich, & Packard 2000)), so the solution is stable.

In the context of the dilemma that arises in neglect-tolerant teaming, Stimpson extended the algorithm further to include more than two actions (by allowing random selection of an action if the aspiration is not met) and more than two players. The latter extension was made by creating a single parameter family of social dilemmas that reduces to a prisoner's dilemma in the two-action, two-player case, but retains the relevant properties of the dilemma for a many-action, many-player case (Stimpson 2002). A complete description of this extension is beyond the scope of this paper, but suffice it to say that the sequential-play prisoner's dilemma abstraction can be caste into this form.

This means that if robots begin with high aspirations and slowly relax these aspirations while randomly trying various policies then they will, in effect, negotiate a situation-dependent, efficient solution to the neglect-tolerant teaming dilemma. Unfortunately, such a solution may take a very long time to achieve. Since there will be many world complexities and human workloads, the algorithm in the current form will be too slow to use in practice. However, the notion of a relaxation search may be very useful in obtaining an efficient solution to the dilemma.

BEANS

The BEANS[5] algorithm is a utility-based algorithm inspired by particle filtering. It is essentially a technique for assigning credit to different approaches to solving a problem; approaches that work well are given high utility. Unlike the relaxation algorithm described above which slowly relaxes aspirations until a solution policy is found with high utility, the BEANS algorithm instead shifts utility between various solution policies until an efficient algorithm is found.

Each algorithm (agent) is composed of *experts* which help it decide what to do in each state. Each expert seeks to fulfill one of the agent's desires, such as get good rewards now, get good future rewards, avoid bad rewards, etc. Experts act by giving support for various policies which they feel will help the agent achieve the desire assigned to that expert. Support for a policy is given by putting“beans” into bins; i.e., a fixed amount of utility units is created, and these units are shifted between different policies according as the experts assign credit and blame. On each trial through the maze in Figure 11 from start to finish, the policy corresponding to the bin with the greatest number of utility units (beans) is used by the agent to choose its action.

After an iteration is completed, each expert is given a certain number of utility beans to allocate to the various policies. In this way, rewards are propagated back through each expert (acting as a learning critic) to the policy that matched the experts desires. Additionally, depending on the rewards received at the end of the iteration, trust in an expert can be increased or decreased, based on whether an expert supported actions which helped (or hurt) the agent. This trust measure affected how many beans are given to an expert on subsequent iterations.

The BEANS algorithm was applied to the sequential prisoner's dilemma described above. Solution strategies were identified from the following list of policies (obtained from multi-agent choice literature but not elaborated upon herein for the sake of space):

- Nash equilibrium
- Pareto optimal
- Always cooperate
- Tit-for-Tat
- Minimax
- Satisficing

Each policy was assigned utility beans by experts that were responsible for the following specific goals:

1. Myopically try to maximize payoff in this round by rewarding a policy if it passes through the bottom door.
2. Avoid bad payoffs by rewarding policies that do not go through one of the top two gates.
3. Reward policies if they produce satisficing behaviors.
4. Maximize a social welfare function that includes the path length of the other agent too.
5. Reward policies that tend to entice the other agent to cooperate in future iterations.

[5]We want to create a clever acronym for this, but have not succeeded yet. The name of the algorithm is inspired by the way utility units are shifted between various policies. This movement can be thought of by a metaphor of moving beans from one bin to another according to some rule.

Interestingly, these heuristic experts usually produced utility patterns that selected mutual cooperation wherein both agents consistently passed through the second to bottom gate. Furthermore, when the algorithm is assigned to one agent and the other agent receives directions from a human, the BEANS agent tends to cooperate when the human cooperates, but adapts to defection when the human defects.

One limitation of this approach is that it requires the *a priori* identification of set of possibly useful policies. This means that the naive application of the algorithm will be unacceptable for many multi-robot domains because the number of agents, the state of the human manager, and the complexity of the world may require the discovery of new policies.

Future Work

This brief presentation of the two algorithms is insufficient for true validation. Much future analysis needs to be done. However, in the abstracted extensive form prisoner's dilemma, both approaches can produce efficient solutions. We conclude that both algorithms deserve future development to try and adapt them to complicated robot team scenarios.

Discussion

This paper has presented the following:

1. a review of recent results from neglect tolerance studies of various interaction schemes,
2. an abstraction of how teams of semi-autonomous, semi-independent robots form a type of multi-agent social dilemma,
3. a description of why such teams should avoid Nash equilibria solutions, and
4. a description of two preliminary mechanisms for having robots *learn* to choose neglect-tolerant autonomy levels in such a way that the human is able to manage multiple robots.

There are a number of issues and observations that arise with such an approach to managing a complex system of multiple robots by having the robots learn neglect tolerant teaming. The first observation is that learning to choose a robot autonomy level in a team context should include estimates of human workload and world complexity. Without such information, it seems likely that learning algorithms will myopically do what they think is best for themselves and thereby produce inefficient team behavior. Indeed, the biggest advantage of using an approach based on machine learning is that extensive neglect tolerance studies need not be done for every interface/autonomy mode pair. Instead, workload and complexity can be used as inputs to a learning algorithm with team performance acting as the feedback that guides adaptation. If the robot learns to cooperate with other robots while accounting form neglect tolerance, then team performance, which benefits from a prudent allocation of human attention, should be high.

To develop algorithms that address neglect-based teaming dilemmas, it is useful to have a test world. We have developed such a world, but there are many flaws with such a world. For example, the world does not address the important question of how to choose the right level of granularity in what a robot should learn (e.g., when to request service, what autonomy level, how much service time to request). Furthermore, it does not address how can we use noisy real-time estimates of human workload and environment complexity (e.g., behavioral entropy (Boer *et al.* 2001)) as states in a learning algorithm. These issues are areas of future research.

References

Axelrod, R. M. 1984. *The Evolution of Cooperation*. Basic Books.

Boer, E. R.; Futami, T.; Nakamura, T.; and Nakayama, O. 2001. Development of a steering entropy method for evaluating driver workload. In *SAE 2001 World Congress*. SAE paper #1999-01-0892.

Crandall, J. W., and Goodrich, M. A. 2001. Experiments in adjustable autonomy. In *Proceedings of Systems, Man, and Cybernetics*. To appear.

Crandall, J. W., and Goodrich, M. A. 2002. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *Proceedings of the 2002 IEEE /RSJ International Conference on Intelligent Robots and Systems*. To appear.

Goodrich, M. A.; Olsen, D. R.; Crandall, J. W.; and Palmer, T. J. 2001. Experiments in adjustable autonomy. In *Proceedings of the IJCAI01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*.

Karandikar, R.; Mookherjee, D.; Ray, D.; and Vega-Redondo, F. 1998. Evolving aspirations and cooperation. *Journal of Economic Theory* 80:292–331.

Pashler, H. E. 1997. *Engineering Psychology and Human Performance*. The MIT Press.

Sheridan, T. B. 1992. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press.

Simon, H. A. 1996. *The Sciences of the Artificial*. MIT Press, 3rd edition.

Stimpson, J. L. 2002. Satisficing solutions to a multi-agent social dilemma. Master's thesis, Brigham Young University, Provo, UT, 84602, USA.

Stirling, W. C.; Goodrich, M. A.; and Packard, D. J. 2000. Satisficing equilibria: A non-classical approach to games and decisions. *Autonomous Agents and Multi-Agent Systems Journal*. To appear.

Wickens, C. D., and Hollands, J. G. 2000. *Engineering Psychology and Human Performance*. Prentice Hall, third edition.