Flexible Multi-Agent Decision-Making Under Time Pressure

Sanguk Noh, *Member*, *IEEE*, and Piotr J. Gmytrasiewicz

Abstract— **Autonomous agents need considerable computational resources to perform rational decision-making. These demands are even more severe when other agents are present in the environment. In these settings, the quality of an agent's alternative behaviors depends not only on the state of the environment, but also on the actions of other agents, which in turn depend on the others' beliefs about the world, their preferences, and further on the other agents' beliefs about others, and so on. The complexity becomes prohibitive when large number of agents are present and when decisions have to be made under time pressure. In this paper we investigate strategies intended to tame the computational burden by using off-line computation in conjunction with on-line reasoning. We investigate two approaches. First, we use rules compiled off-line to constrain alternative actions considered during on-line reasoning. This method minimizes overhead, but is not sensitive to changes in real-time demands of the situation at hand. Second, we use performance profiles computed off-line and the notion of urgency (i.e., the value of time) computed on-line to choose the amount of information to be included during on-line deliberation. This method can adjust to various levels of real-time demands, but incurs some overhead associated with iterative deepening. We test our framework with experiments in a simulated anti-air defense domain. The experiments show that both procedures are effective in reducing computation time while offering good performance under time pressure.**

Keywords— **Agent modeling, real-time decision-making, adaptive agents, urgency, anti-air defense**

I. Introduction

OUR aim is to design autonomous agents which are boundedly rational while operating in multi-agent settings and under time pressure. Using the paradigm of expected utility maximization for rational decision-making in which there is uncertainty is known to be computationally demanding [1]-[10]. We consider additional complexities presented by multi-agent environments. In these settings, the values of alternative courses of action an agent can consider depend not only on the possibly complex and not fully known state of its environment, but also on the actions of the other agents. What the other agents can be expected to do, however, depends on their capabilities, goals, and beliefs. Further, the other agents' beliefs about other agents' intentions may be relevant, as well as their beliefs about others' preferences, beliefs, and so on. Clearly, computational demands may lead to an agent's failure to decide on how to act in an acceptable time.

Our approach is to rely on results accumulated during off-line computation, and to use these results in conjunction with on-line reasoning. First, off-line computation can be used to summarize regularities of decisions made in a sample of randomly generated situations. We chose to represent the found regularities as rules dictating which actions should be candidates for execution and which could be rejected outright, depending on the circumstances. This information can be passed on to the on-line decision making module to limit the number of alternative actions it needs to choose among. For example, in many settings, an agent cooperating with numerous other agents does not need to worry about tasks that are remote from it but close to other agents. These tasks can be usually safely ignored if the other agents are capable of performing them, and the agent can concentrate its computational resources on more relevant alternatives. This method is simple and usually effective but not very flexible. If, say, 10 out of 25 alternatives are judged as implausible, but choosing among the remaining 15 still takes too long the agent is going to fail.

Our attempt to make deliberative reasoning more reactive is similar in spirit to methods advocated in [11], [12]. However, most approaches rely on ready-to-use plans predefined by the system designer and on the rules guiding their execution. Our rules compile the results of off-line deliberations and narrow the set of alternative actions considered by a deliberative component.

Another way of utilizing off-line computation is to randomly generate sample problems and compute optimal decisions off-line with progressively more information. The resulting performance profile summarizes the increase in the quality of decisions made due to increased computational effort. In other words, it quantifies the expected value of computation in the class of sample problems. This value can be compared to the cost of time, computed on-line based on the situation at hand, to determine whether it pays to invest more computation time in solving the problem. This method is more flexible in that it is sensitive to the urgency of the situation at hand; if the current estimate is that time is valuable the agent will not invest more in the computation and execute the best action arrived at so far. The agent is guaranteed not to run out of time if its estimate of the value of time is accurate, and performance is subject to the accuracy of this estimate. We present a method for computing this estimate here.

Using performance profiles has been advocated in previous work [3], [5], [7], [13]. The approach we propose con-

Manuscript received July 21, 2002; revised October 2, 2003. This work has been supported by the Korea Research Foundation under grants KRF-2002-041-D00465 and KRF-2003-041-D20465, and was sponsored in part by the Office of Naval Research Artificial Intelligence Program under contract N00014-95-1-0775 and by the National Science Foundation CAREER award IRI-9702132.

S. Noh is with the School of Computer Science and Information Engineering, The Catholic University of Korea, Republic of Korea (e-mail: sunoh@catholic.ac.kr)

P. J. Gmytrasiewicz is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: piotr@cs.uic.edu)

tributes by defining and using the notion of urgency, or the value of time, and by applying it to multi-agent settings.

The paper is organized as follows. In Section II we briefly compare our approach to related research. Section III briefly presents the domain of simulated anti-air defense and the Recursive Modeling Method, a deliberative decision-making framework for multi-agent settings. Section IV is devoted to methods that compile regularities of the air-defense domain into rules that constrain alternative actions that need to be considered. For example, an agent may be able not to worry about tasks if the chances that another agent will be able to take care of this task are sufficiently high. In Section V we derive a method that uses performance profiles and an estimate of the value of time to optimally trade-off cost and benefit of further computation. Section VI provides a series of experiments showing the performance of agents using the developed procedures in the time-critical, anti-air defense domain. The experimental results show that our approaches enable rational agents to function and fulfill their mission under time pressures. In the concluding Section VII, we summarize our results and discuss further research issues.

II. Related Work

Our work builds on efforts by several other researchers who focus on making autonomous systems practical while functioning in complex environments and under time constraints.

One approach is to use a purely reactive reasoning procedure [14]-[16]. The goal of reactive systems is to respond to an observed condition with a predefined action. Although such a system may be able to react very quickly to environmental conditions, predefined plans are of less value if a situation changes and re-planning is needed. Further, in multi-agent settings, if no protocol or the overall plan can be established explicitly in advance, a reactive system is likely to lock the agents into suboptimal forms of behavior. Our flexible reasoning procedures provide for bounded rational behaviors in dynamic environments when plans and protocols have not been defined in advance.

In another approach, Bratman et al. [11] describe an agent's architecture that includes both means-end reasoning and decision-theoretic reasoning. For a resource bounded agent, the agent's beliefs, desires, and intentions (BDI) are involved in the formation, revision, and execution of plans to constrain the deliberation process. Rao et al. [12], [17] explore combining reactive and deliberative behaviors provided by the BDI architecture, and use it for a air-traffic management system OASIS. They advocate that the agent commit to one of its plans and periodically reconsider its intentions, but the issue of how often, and based on what information, to reconsider is left open.

Other rigorous efforts have focused on the meta-level rationality [1], [2], [18] of deliberative agents, which takes into account deliberation costs, i.e., computation time used for deliberation (see [13] for detailed summary). The decision-theoretic agents following the meta-reasoning principle calculate the expected improvement of decision quality given the investment of computation time. These approaches include work on the use of meta-reasoning procedures to control inference [7], [19], and anytime algorithms [5], [9]. Meta-reasoning and anytime algorithms balance the costs and the benefits of continued deliberation [2], [13] by performing online computation of expected utilities of additional deliberation. Schut and Wooldridge [20] apply the model of meta-reasoning to BDI agents to decide whether or not intentions are reconsidered. Our approach also takes the perspectives of meta-reasoning and anytime decisions in that both approaches return different qualities of resulting behaviors with different amounts of computation, and incorporate off-line simulation data into performance profiles. The contribution of our work is to precisely define the notion of the value of time, or urgency, and to use it in conjunction with performance profiles in multi-agent settings.

In the field of multi-agent coordination, there have been several approaches to support practical decision-making. Lesser [21] proposes a domain independent agent architecture for a large-scale multi-agent environment, which deals with adaptation to resource availability, trade-offs between resources and quality, and complex interdependencies among agents. The implementation of components of their software infrastructure and testing in sophisticated, real-time multi-agent applications is an ongoing work. Durfee [22] suggests strategies for making coordination practical when agents use the Recursive Modeling Method [22]-[25], which we also use, as a decision-making framework. These strategies include simplifying utility computations, limiting possible choices, treating multiple agents as one, and so on. This work is similar to our current work and we build these ideas, but we put them on the more formal footing of defining precisely the urgency of the situation, computed on-line, and the benefits of continued computation, prepared off-line.

III. Deliberative Decision-Making in Anti-Air Defense

Our framework for rational decision-making in multi-agent settings is based on the Recursive Modeling Method (RMM) [22]-[25]. In RMM an agent chooses its action so as to maximize its expected utility based on what it expects the other agents' choices will be. The expected actions of others are derived from the models of the other agents' decision-making, as depicted in Fig. 1. The fact that the other agents might themselves be modeling others gives rise to a recursive hierarchy of models called a recursive model structure (RMS) (we give a concrete example of decision-making using RMM later in this section).

In [25] the models are formalized as payoff matrices, but they can also take the form of influence diagrams. To learn the models, agents can use forms of Bayesian learning based on observations of others, as we described in [24]. In [25] we assumed that the agents can only possess finitely nested information about each other, which leads to recursive model structures of finite depth. Such finite recursive model structures can be solved using dynamic programming, but, in

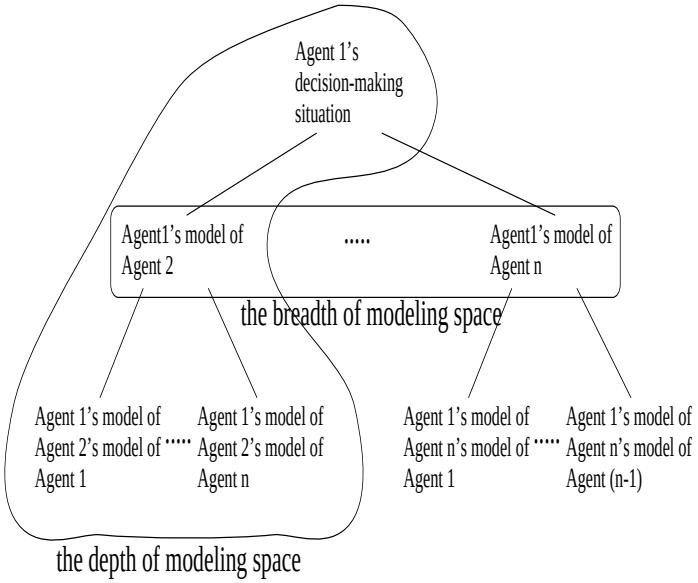


Fig. 1. A recursively nested models of agents comprising the recursive modeling structure (RMS) of Agent1.

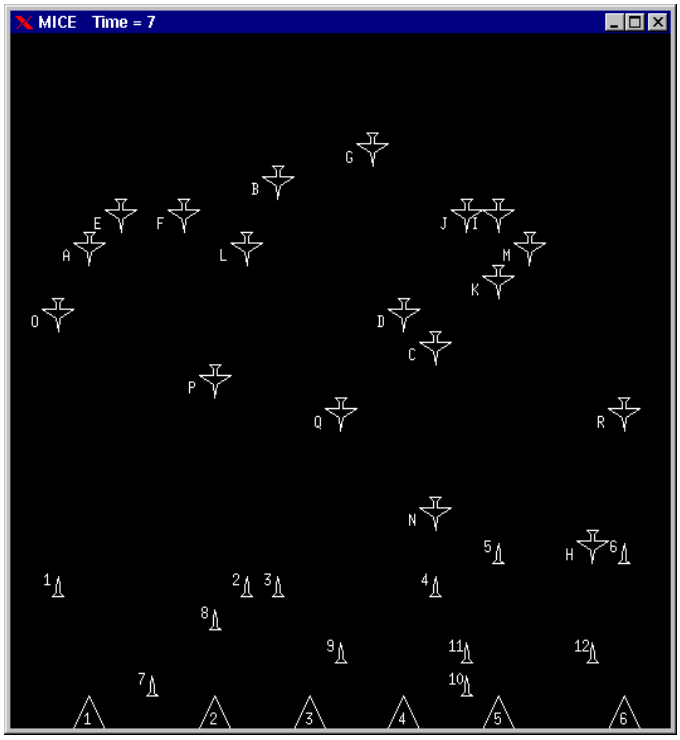


Fig. 2. A complex anti-air defense scenario. In this scenario, the targets are labeled A through R, the defense units are 1 through 6 within the triangles, and the interceptors are 1 through 12.

the worst case, the size of the structure is still exponential in the number of agents. It is clear that the purely deliberative, full-blown RMM needs to be supplemented by other, more reactive methods for more complex and time-critical scenarios [26].

Let us take a specific example of anti-air defense domain [27]-[30]. It consists of a number of attacking targets and coordinating defense units, as depicted in Fig. 2. This scenario has 18 incoming targets, labeled A through R, and 6 defending agents, labeled 1 through 6, depicted as triangles on the bottom of the figure. Each of the defense units are assumed to be equipped with interceptors, and to be capable of launching them, if they are not incapacitated. The defense agents autonomously decide to launch their interceptors, labeled 1 through 12 in Fig. 2, at selected targets. The incoming targets keep moving straight down to attack the territory on which the defense units are located.[1] The mission of the defense units is to attempt to intercept the attacking targets so as to minimize damages to their own territory. Let us note that our model promotes coordination – if the defense agents mis-coordinate and attempt to intercept the same threat, another threat could reach its destination and cause damage proportional to the warhead size.

The defense agents analyze a particular defense situation in decision-theoretic terms by identifying the attributes that bear on the quality of alternative actions available to the agent. First, each attacking target presents a threat of a certain value, here assumed to be the size of its warhead. Thus, the defense units' preference is to intercept threats that are larger than others. Further, the defense units should consider the probability with which their interceptors would be effective against each of the hostile targets. The interception probability, $P(H_{ij})$, depends on the angle, γ_{ij}, between the target j's direction of motion and the line of sight between the battery i and the target (see, for example, [31]), as follows:

$$P(H_{ij}) = e^{-\mu\gamma_{ij}} \tag{1}$$

where μ is an interceptor-specific constant (assumed here to be 0.01).

The fact that the incoming targets travel down and that it becomes more difficult to intercept them as time proceeds means that time is valuable, and the situation is urgent. We define the notion of urgency, or value of time, below, and our agents compute it on-line to limit their computation.

To represent agent's beliefs about the world and other agents, we implemented an agent's knowledge base (KB) as a hierarchical system of classes and objects which are recognized in the environment [29], [32]. As the agent identifies new objects in its environment, it creates instantiations of classes in its KBs to represent information about these objects. As we mentioned, the agents compile the information contained in their declarative KB's into compact payoff matrices to facilitate expected utility calculations. A payoff matrix [33], [34] used in game theory can be seen to faithfully summarize the information contained in the KB by listing the expected payoffs of possible decisions, depending on the parameters (attributes) describing the domain.

For the sake of simplicity we will explain our modeling approach in a much simpler case of two targets and

[1]Several demonstrations of the air defense domain are located at http://dali.ai.uic.edu.

two defense agents. We'll assume that each of the batteries has only one interceptor (we conducted experiments in more complicated settings involving multiple salvos and more attacking targets as we explain later on). In a simple anti-air defense domain, shown in Fig. 3, the top left corner of the screen is assumed to have coordinate value of $(0, 0)$, with x pointing right, and y pointing down. The expected benefit of firing an interceptor at a threat can be quantified as a product of the size of the threat, i.e., its warhead size, and the interception probability. For example, in Fig. 3, the expected utility of Battery1's shooting at the threat A and Battery2's shooting at the threat B amounts to $206.8 (= 100 \times 0.94 + 120 \times 0.94)$. This value is entered in the payoff matrix, such as one on top in Fig. 4. In this payoff matrix the rows represent Battery1's alternative actions of shooting at A, B, and not shooting at all (S), respectively, and the columns represent the alternative actions of Battery2.

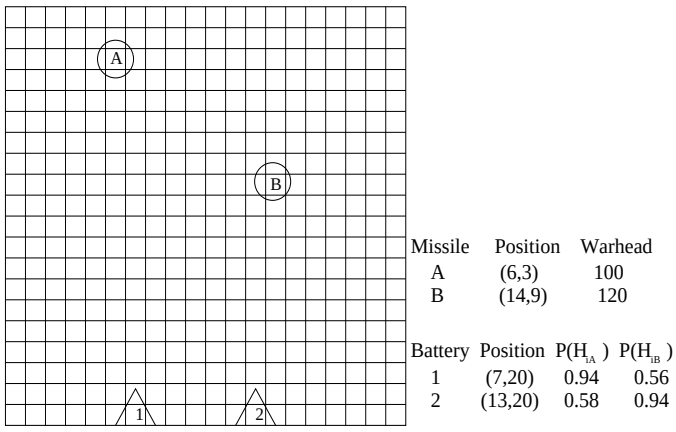


Fig. 3. An example of a simple anti-air defense scenario.

In Fig. 4, the top payoff matrix compactly represents the decision-making situation that Battery1 is facing. To solve it, Battery1 needs to predict what Battery2 is likely to do; for example if Battery2 were to shoot at A it would be best for Battery1 to shoot at B, but if Battery2 is expected to shoot at B then Battery1 should definitely target A.

The actual behavior of Battery2, however, depends on a number of factors that Battery1 may be uncertain about. Specific values of these factors lead to a number of alternative models of Battery2's decision-making situation. Each model is used to probabilistically predict Battery2's behavior. For example, if Battery2 has been hit and incapacitated, it will not be able to launch any interceptors. This model can be represented by the probability distribution of $[0, 0, 1]$, which assigns the probability of 1 to Battery2 not shooting, and zero to its shooting at A and B.

If Battery2 is not incapacitated and it has long range interceptors (suitable only for intercepting target A) and short range interceptors (suitable only for B), then its decision-making situation can be represented as another payoff matrix. This is the left-most matrix in the second level of structure in Fig. 4. Further, Battery2 may have run out of long range or short range interceptors. If Battery2 has only long range interceptors, it would be unable to attack target B, and can only attempt to intercept A. If Battery2 has only short range interceptors, it can only attempt to shoot at target B. These four models, of Battery2 being fully operational and having long and short range interceptors, operational with only long-range interceptors, having only short-range interceptors, and incapacitated, are depicted as the second level models in Fig. 4, with their associated probabilities, in this case 0.3, 0.3, 0.3, and 0.1, respectively. The probabilities associated with the alternative models can be arrived at using Bayesian update based on the observed behavior of the other agents, as we described in [24], [29].

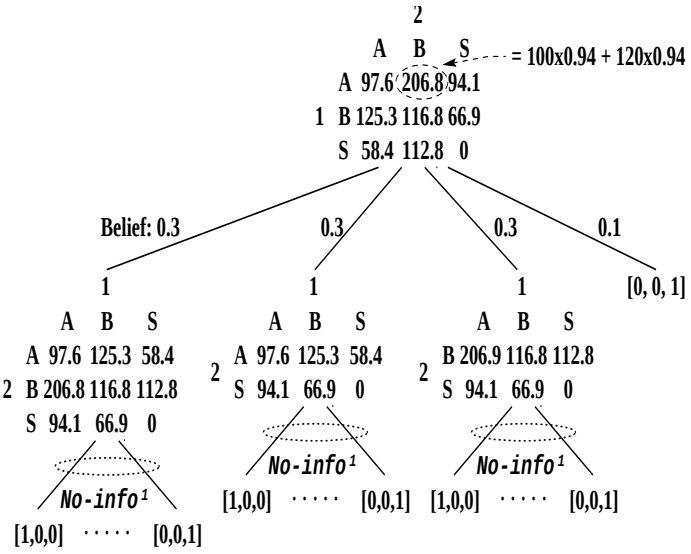


Fig. 4. RMM's recursive model structure for Battery1's decision making.

The resulting hierarchy of models terminates with a *No-info* model when the agent (in this case Battery1) runs out of modeling information. The recursive modeling could continue into deeper levels, but in this case we assumed that Battery1 has no further information about how Battery2 is modeling Battery1. Since models are used to estimate behavior, the *No-info* model in this case represents a uniform probability distribution over the space of all possible probabilistic predictions Batter2 might have as to Battery1's behavior (see [25] for details). We have designed and implemented a numerical method of solving the *No-info* models [27], [28] using logical sampling.

The Recursive Modeling Method uses dynamic programming bottom-up solution [27], [28] to process model structures as in Fig. 4 and determine the rational choice of coordinated action. In this case, Battery1 computes that if Battery2 is fully operational then the probability distribution over Battery2's actions A, B, and S is $[0.03, 0.97, 0.0]$. We used the sampling algorithm with the density of 0.1 to account for *No-info* below in the model structure. If Battery2 has only long-range interceptors it will choose to shoot at target A, i.e., the probability distribution over Battery2's actions becomes $[1, 0, 0]$. If Battery2 has only

short-range interceptors it will intercept target B. These probability distributions are combined using the beliefs of the alternative models:

$$0.3 \times [0.03, 0.97, 0] + 0.3 \times [1, 0, 0] + 0.3 \times [0, 1, 0] + 0.1 \times [0, 0, 1] = [0.31, 0.59, 0.10].$$

The resulting distribution is Battery1's overall expectation of Battery2's actions, given all of the remaining uncertainties. Propagating these results to the upper level, the combined probability distribution describing Battery2's actions is used to compute the expected utilities of Battery1's alternative actions. We have

$$EU(A) = 0.31 \times 97.6 + 0.59 \times 206.8 + 0.10 \times 94.1 = 161.74$$
$$EU(B) = 0.31 \times 125.3 + 0.59 \times 116.8 + 0.10 \times 66.9 = 114.45$$
$$EU(S) = 0.31 \times 58.4 + 0.59 \times 112.8 + 0.10 \times 0 = 84.66.$$

Thus, given the uncertainties about Battery2, Battery1's rational coordinated choice is to intercept target A. Note that if Battery1 had more modeling information, for example if it was known how Battery2 is modeling Battery1, the estimates above would be more precise, and the expected utility of the best action would be higher. As expected, additional information improves the quality of the result, but at the expense of computational effort needed to process it. The improving quality of the choice made, the additional computation time, and the urgency of the situation are the three factors we use to bound the rationality of an agent operating under time pressure.

IV. Compilation of Decision-Making into Rules

The first of the decision-making procedures we consider uses compiled rules to constrain the set of alternative actions among which the agent needs to choose. This helps to constrain the computational effort, since a smaller number of alternative actions translate into a smaller payoff matrix that the agent has to process.

The rules are constructed by compiling the deliberative decision-theoretic reasoning. The compilation process exploits the regularities of the multi-agent domain and avoids time consuming deliberations in urgent situations. We use machine learning algorithms to obtain the rules. For input cases, we use the results of decision-theoretic solutions of full-blown recursive model structures performed off-line in randomly generated anti-air defense settings. The various compilations available, and their combinations with deliberative methods, constitute a spectrum of approaches to making decisions under the constraints of available computational and cognitive resources, and under time pressure. We chose an architecture according to which rules obtained off-line serve as a pre-processing stage to the more accurate but slower deliberative decision-making. In other words, the compiled rules filter out the actions which are likely to be suboptimal, and pass only the plausible actions to the next step of the deliberative reasoning.

We will use the following notation:

- a set of agents: $N = \{n_1, n_2, \ldots\}$;
- a set of actions of agent n_i, $n_i \in N$: $A_{n_i} = \{a_i^1, a_i^2, \ldots\}$;
- a set of possible world states: $S = \{s_1, s_2, \ldots\}$;
- a set of states of knowledge of agent n_i: $KB_i = \{kb_i^1, kb_i^2, \ldots\}$;
- a set of compilation methods, $L = \{l_1, l_2, \ldots\}$, corresponding to machine learning algorithms employed by the agent.

The expected utility of the best action, α, of agent n_i, arrived at using the body of information E, and executed at time t, is given by[2]

$$EU(\alpha|E,t) = \max_{a_i^j \in A_{n_i}} \Sigma_k P(s_k|E,t,a_i^j) U(s_k) \qquad (2)$$

where

- $P(s_k|E,t,a_i^j)$ is the probability that a state s_k will obtain after action a_i^j is executed at time t, given the body of information E;
- $U(s_k)$ is the utility of the state s_k.

An agent which is not subject to computational limitations would substitute its full state of knowledge for E above, and compare expected utilities of all possible actions immediately. The impracticality of this computation motivates us to consider the rules, which limit A_{n_i}, and the reasoning about the scope of modeling knowledge that should be used, which limits E.

Since the actual state of the world may not be fully observable, elements of S cannot be used as a left-hand-side of a rule. Instead, the antecedents of the rules are descriptions of the agent's KB expressing what is known, as well as what is uncertain and to what degree. This leads to rules that could specify, for instance, whether an incoming target located at the other end of the defended territory does, or does not, have to be considered as a plausible choice, depending on the probability that the agent defending that part of the territory is incapacitated. We would expect that, if it is known that the other agent is fully operational, the learned rules would allow the agent not to consider the target. If, however, the target size is large enough, and the uncertainty of the other agent being operational is high enough, the threat will be included among ones considered for interception.

Given a learning method $l_m \in L$, a set of compiled decision-making rules of agent n_i is defined as

$$\omega_{l_m}^i : KB_i \times A_{n_i} \mapsto \{\textsc{True}, \textsc{False}\} \qquad (3)$$

representing that an action should be considered (True), or not be considered (False), in an arbitrary condition of the agent's n_i state of knowledge.

Given a specific state of knowledge kb_i^k and a learning method l_m, the set of plausible actions A'_{n_i} for the agent n_i is

$$A'_{n_i} = \{a_i^j : a_i^j \in A_{n_i} \text{ and } \omega_{l_m}^i(kb_i^k, a_i^j) = \textsc{True}\} \qquad (4)$$

[2]Our notation follows [3], [35].

so that $A'_{n_i} \subseteq A_{n_i}$. The set of plausible actions results from applying the compiled rules in the $\omega^i_{l_m}$ to the current state of knowledge and all possible alternatives. Say, given kb^k_i and $A_{n_i} = \{a^1_i, a^2_i, a^3_i\}$, if $\omega^i_{l_m}(kb^k_i, a^1_i) = \text{TRUE}$, $\omega^i_{l_m}(kb^k_i, a^2_i) = \text{FALSE}$, and $\omega^i_{l_m}(kb^k_i, a^3_i) = \text{TRUE}$, then a^1_i and a^3_i are plausible, and a^2_i is not, according to the rule set $\omega^i_{l_m}$. Given the set of pre-filtered plausible actions our agent maximizes the expected utility among them [3], [35] so its best action is

$$\arg\max_{a^j_i \in A'_{n_i}} EU(a^j_i|E,t). \quad (5)$$

Various machine learning algorithms compile decision-theoretic models into different functions ω_{l_m}, each enabling the agent to take advantage of regularities in the environment in a different way. As mentioned, we generate the training examples for the learning algorithms from the results of deliberative reasoning performed by the Recursive Modeling Method in randomly generated air defense episodes. We discuss specific results below, after we introduce the other flexible decision-making procedure.

V. Flexible Decision-Making Using the Value of Time

As we mentioned, in multi-agent domains, the difficulty lies in computing $P(s_k|E,t,a^j_i)$ because these transition probabilities depend also on the actions of other agents, which have to be included as conditioning terms. Our approach is based on expected utility maximization when the information, E, is contained in Recursive Modeling Method (RMM)'s recursive model structure.

The procedure we consider now is aimed at identifying the appropriate scope of modeling, E, based on performance profiles [5], [8], [9] computed off-line, and on the urgency of the situation [3]-[5], [7], [8] computed on-line. We are inspired by techniques used by humans in real-life situations when they make coordinated decisions with a large number of individuals. We frequently consider and think about only a few agents in our immediate vicinity for the purpose of coordination. Similarly, our agents begin by considering their closest neighbors first, and then broaden the scope by including more distant agents (in Fig. 1, this corresponds to broadening of the recursive modeling structure). At the same time, our agents need to decide on how deep the modeling should proceed. While reasoning with information at deeper levels, if available, results in better decisions, it also requires more computation time.

Given the dimension of breadth, which corresponds to the number of agents modeled, and depth, which corresponds to the reasoning level in the recursive model structure, as depicted in Fig. 1, we use an *iterative deepening algorithm*. The iterative deepening algorithm starts with the least amount of information about the world and other agents, and decides on a default action which is ready to execute almost immediately. Including more information, by increasing the depth and breadth of the recursive model structure, results in actions with higher expected utility. The actual quality of the resulting decisions can be tested in randomly generated interaction episodes. The resulting increase in performance is represented as a performance profile [13], [18].

Further, we define the notion of urgency, or the value of time, computed on-line given a situation at hand. The intention is to enable the agent to be optimal as it trades off the value of time for increase in quality of more informed decision-making. Thus, if time is valuable an agent is predisposed to arrive at a decision quickly. When the situation is not urgent (the time is not very valuable), the agent can afford to consider more information and spend more time deliberating and looking for better solutions.

Definition 1: Given a state of knowledge, kb_i, of an agent, n_i, the urgency, or the value of time to n_i, is the rate of disutility accumulation due to n_i's inaction.

Our definition above is closely related to a notion called opportunity cost in the jargon of the economics literature.

We now have defined all of the elements allowing optimal trade-off of computation and time. To derive how these notions should be combined let us consider an agent which arrived at the currently favored action, α, using body of information E. Agent n_i can execute the action immediately, or continue computing by increasing the body of knowledge considered, and including additional information, e [3], [5], [7]. If the agent were to continue computing with information $E+e$, the result would be α'. The new decision could be the same as α, and would be available for execution at time $t+t'$, where t' is the computation time used. The decision of whether to keep computing with extra information or whether to execute the best current action comes down to following comparison:

$$EU(\alpha'|E+e, t+t') > EU(\alpha|E,t). \quad (6)$$

That is, if the expected utility of delayed action arrived at using more information is greater than the utility of acting now on the current best, the agent should keep on computing using the information $E+e$. The challenge is to determine whether the above inequality holds *without* actually computing $EU(\alpha'|E+e, t+t')$. Using performance profiles one can estimate $EU(\alpha'|E+e,t)$, i.e., the expected value of decision based on information $E+e$, executed currently. Since the actual action α', based on the information $E+e$, is not available, we postulate that the currently available α be used to estimate the $EU(\alpha'|E+e, t+t')$ as follows:

$$EU(\alpha'|E+e, t+t') \approx EU(\alpha'|E+e,t) - [EU(\alpha|E,t) - EU(\alpha|E,t+t')]. \quad (7)$$

Thus, the value of delayed action α', which is currently not available, is computed assuming that its quality deteriorates with time at the rate equal to that of α, which has

already been computed. Substituting the above into (6) and rearranging terms, we get

$$EU(\alpha'|E+e,t) - EU(\alpha|E,t) > EU(\alpha|E,t) - EU(\alpha|E,t+t'). \quad (8)$$

The left hand side of the above inequality is the benefit of further computation which can be estimated from the performance profile in case that the best current action can be carried out instantaneously. The right hand side is the current estimate of urgency, or the value of time, defined before as the rate with which the expected utility of the best current alternative deteriorates with time. The trade-off between the value and cost of further computation is now clearly defined. For example, if the system slowly processes the data and the urgency is high, then the value of time is greater than the benefit of computation, and immediate action should be taken.

VI. Testing and Evaluation

The experiments in this section are designed to validate the rational coordination and flexible decision-making procedures by applying them to anti-air defense domain. We use an anti-air defense simulator written in Common LISP and built on top of the MICE simulator [36], running on a LINUX platform. For each scenario in the simulated environment, the positions and warhead sizes of targets are randomly generated. The target positions range from 0 to 10 and the warhead sizes vary between 100 and 300.

We use the following settings:

- *2 vs. 2.* There are two defense agents and two incoming targets. The 2 vs. 2 setting is for preliminary learning of rules that limit the number of alternative behaviors;
- *6 vs. 6.* There are six defense agents and six incoming targets. This setting is used to generate performance profiles and to test the calculation of urgency;
- *6 vs. 18.* There are six defense agents and 18 targets. The 6 vs. 18 setting is used to test the two procedures we considered in scaled-up situations.

We measure an agent's performance in terms of the total threat removed. The *total threat removed* is a sum of the removed warhead sizes of the attacking targets. If a target was not intercepted it does not contribute to the performance at all. The agent's performance and computation time were gathered on a 500 MHz Pentium^{TM} III single-CPU machine.

A. Compilation of Deliberative Decisions

To construct compiled rules for our agents in the coordinated defense domain we used four machine learning algorithms: naive Bayes classifier[3], C4.5 [38], CN2 [37], and FOIL [39]. Hence, the set of compilation methods L was {Bayes ($= l_1$), C4.5 ($= l_2$), CN2 ($= l_3$), FOIL ($= l_4$)}. For the Bayesian classifier, the results are represented as rules specifying the probability of occurrence of each attribute value given a class [37], in our case True (also called *Select Target* below) and False (also *Don't Select Target* below). C4.5 represents its output as a decision-tree, and the output of CN2 is an ordered set of if-then rules. The trained results of FOIL are the relations of attributes as function-free Horn clauses. We now describe the agent's compiled knowledge by using the above learning algorithms, and compare their decision capabilities in the anti-air defense environment.

[3]We implemented a simple Bayesian classifier described in [37].

A.1 Learned Condition-Action Rules in 2 vs. 2 Settings

In our experiments we considered agents that vary in their capacities, and that have limited knowledge about other agents. Some of the attributes contained in agents' KBs when they interact in an anti-air defense environment are summarized in Table I. They include the size, speed and angle of the attacking targets, the agent's own intercepting capabilities (i.e., its possessing both long- and short-range interceptors, only long-range interceptors, only short-range interceptors, and its being incapacitated and unable to shoot), and the probabilities associated with the capabilities of the other defense agent[4] (the other agent's possessing both long- and short-range interceptors, only long-range interceptors, only short-range interceptors, and its being incapacitated and unable to shoot).

TABLE I
The relevant attributes in the *2 vs. 2* anti-air setting

Attribute	Type	Value
Target Size	numeric	100 - 300
Target Speed	discrete	slow, mid, fast
Target Angle	numeric	0 - 90
Distance	numeric	0 - 20
Capacity	discrete	both, long, short, incap.
P(long, short)	numeric	0.0 - 1.0
P(only long)	numeric	0.0 - 1.0
P(only short)	numeric	0.0 - 1.0
P(incapacitated)	numeric	0.0 - 1.0

Given a scenario composed of the attributes in Table I, our agents made deliberative decisions using the Recursive Modeling Method, as described in Section III. The resulting decisions were fed into the learning algorithms as training data. For each target in an anti-air defense setting, the training data was obtained as a tuple of attribute values and a class True, if the target is selected, or False, otherwise. The learning algorithms l_1 through l_4, then compiled the training data into a different set of rules ω_{l_m}. The learned decision rules ω_{l_m}, as defined in (3), take into account only the properties of one target at a time, not of the whole configuration. As an example, a decision tree obtained using C4.5 ($= l_2$) for these attributes is depicted in Fig. 5. Based on the resulting decision tree, one of compiled rules in ω_{l_2} is "if Capacity = short and Distance >= 15, then False."

[4]Table I describes parameters for two agents and the generalization to a set of agents requires an additional set of probability parameters.

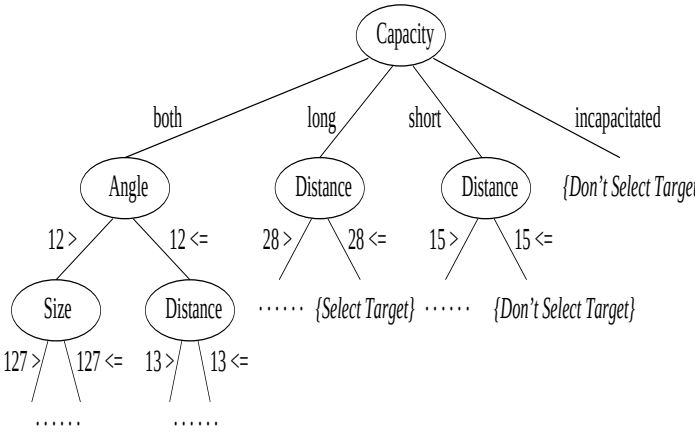


Fig. 5. The decision tree obtained by C4.5.

A.2 Performance of Compiled Rule Sets

To evaluate the quality of various rule sets generated by different learning algorithms the performance obtained was expressed in terms of the total threat removed from friendly forces after launching interceptors.

To guarantee the soundness of the learning hypothesis, we generated several sets of training examples. As the number of the examples increased, the resulting performance improved sharply up to a certain point, after which performance leveled off. We found that, in anti-air defense scenarios that included two batteries and two targets, the sufficient number of training instances was 250.

We applied the compiled condition-action rules ω_{l_m}, obtained by different learning methods l_m, into a new set of 250 scenarios, and could get the set of plausible actions A'_{n_i}, as described in (4). We tested the performance of the reactive learning methods as well as that of the deliberative RMM. The results of average performance (the *total threat removed*) and computation time (*second*) are described in Table II.

TABLE II
PERFORMANCE AND RUNNING TIME OF ALGORITHMS IN THE TWO DEFENSE AGENTS AND TWO TARGETS SETTING

Methods	Performance	Computation time
Deliberative		
RMM	200.6 ± 97.9	0.0062 ± 0.0048
Reactive		
Bayes	170.5 ± 105.9	0.0027 ± 0.0044
C4.5	158.4 ± 102.2	0.0019 ± 0.0039
FOIL	155.9 ± 104.8	0.0024 ± 0.0042
CN2	154.3 ± 100.5	0.0025 ± 0.0043
ANOVA	8.935	15.592

We analyzed the performance results in table II using the standard analysis of variance (ANOVA) method. Since the computed value of $f = 8.935$ in ANOVA exceeds $3.32 (= f_{.01,4,\infty})$ from the F distribution, we know that the five teams consisting of two defense agents controlled by five different methods were not all equally effective at the 0.01 level of significance (i.e., the differences this large in our sample would only arise with a probability of less than 0.01 if there were no real differences). In Table II, the obvious difference was between the deliberative RMM and the sets of reactive rules. The average performance of RMM was 200.6 while that of reactive methods was 159.8, and the average computation time of RMM was 0.0062 while that of reactive methods was 0.0024.

As we expected, the deliberative RMM showed the best performance. The naive Bayesian classifier computed the probabilities for two classes, and enabled the defense agents to select the target which had the highest probability of being in the class of targets to be intercepted. Due to containing information about the probabilities for each target, the naive Bayesian classifier showed both good performance and relatively short running time. When the rule sets obtained from C4.5, CN2, and FOIL were used, they frequently, and somewhat uniquely, were unable to decide which target to intercept in some cases. In other words, both targets in the setting were classified into the set of plausible actions, or both targets frequently were not recommended for interception. If the defense agents still were ambiguous in target interception after applying compiled rules, they randomly selected the target, which explains poor performance.[5]

RMM required the longest running time, and C4.5 needed the shortest running time among the five decision procedures. ANOVA revealed that the five running times were not all equally effective at the 0.01 level of significance. When making a decision, the defense agents compared the current sensory input with one of the condition-action rules to classify a new example. The learning results obtained from CN2 and FOIL were represented by the sequential comparisons of if-then clauses while C4.5 used the decision tree, which computes the entropy of each attribute. Due to this difference it took longer to run CN2 and FOIL than C4.5. The advantage of the decision tree was in that it reduced the number of matches by disregarding nodes in the tree unrelated to the input.

To measure the efficiency of an agent which uses the reactive rules to filter the alternatives passed on to a deliberative procedure, we experimented with the scaled-up setting depicted in Fig. 2. In this setting, there were six defense agents and 18 incoming targets. The RMM agents we implemented for these experiments modeled only their closest neighbors for coordinated decision-making to reduce their deliberation time. Since, as shown in Table II, the Bayes classifier and C4.5 performed best among the reactive rule sets in the 2 vs. 2 setting, we used these two learning algorithms to get a set of plausible actions for our agents in the scaled-up setting. As training data for the two learning algorithms, we generated $13,800$ tuples, which consist of $10,200$ for *Don't Select Target* class, and $3,600$ for *Select Target* class. The results of average performance (the *total threat removed*) and computation time (*second*) in the scaled-up setting are described in Table III.

[5]This alone suggests that the rules are useful as filters but not as ultimate decision tools, which is what we tested for further below.

TABLE III
PERFORMANCE AND RUNNING TIME OF ALGORITHMS IN THE SCALED-UP SETTING

Methods	Performance	Computation time
Deliberative		
RMM	2458.5 ± 261.8	18.455 ± 1.708
Compilation		
Bayes-RMM	2389.1 ± 259.7	2.075 ± 2.229
C4.5-RMM	2361.1 ± 233.2	1.004 ± 0.653
Reactive		
Bayes	2164.9 ± 267.4	0.791 ± 0.043
C4.5	1885.5 ± 236.3	0.127 ± 0.009

Table III presents the average total threat removed after 200 randomly generated defense episodes. We focus on the performances of three different agents: RMM, Bayes-RMM, and C4.5-RMM. The Bayes-RMM agents use naive Bayesian classifier to generate a set of plausible actions (A'_{n_i}), and then using RMM to decide among the plausible actions, as expressed in (5), and similarly for C4.5-RMM. The performance of the purely deliberative RMM agent was, again, the best. The performance levels of Bayes-RMM and C4.5-RMM agent were 97.2% and 96.0% of the RMM agent's performance, respectively. The average size of the set of filtered plausible alternative actions ($|A'_{n_i}|$) were 7 and 9 out of 18 targets for Bayesian classifier and C4.5, respectively.

The running times of agents with filtered actions were drastically reduced. The total running time for Bayes-RMM agent was 2.075 on the average, and the C4.5-RMM agent needs 1.004 seconds for its target selection in the simulated anti-air defense environment. This result indicates that the combination of reactive and deliberative decision-making procedures saves the agent's deliberation time while offering good performance, compared with purely deliberative procedure. As before, among the purely reactive procedures, the performance of Bayesian classifier was better than that of C4.5 because it included the additional probabilistic information that the agent put the plausible targets in order of their probabilities within the "select for interception" category. Since the defense agents controlled by C4.5 alone randomly selected the targets after filtering out unprofitable targets, its performance was the worst.

B. Performance Profiles of Iterative Deepening Algorithm

Performance profiles using the iterative deepening algorithm was generated in the six defense agents and six incoming targets setting, as depicted in Fig. 6. In the experiment, the breadth of modeling space ranges from two to six and the depth of modeling space varies between one and four (see Fig. 1). The algorithm begins with the simplest modeling space, that is, two agents and reasoning depth one and incrementally adds the amount of information up to six agents and depth four. For baseline performance, we include the performance of agents controlled by *random* and *independent* (i.e., without modeling other agents) target selection strategies. The experimental results of average performance (the *total threat removed*) and computation time (*second*) after 100 runs are summarized in Table IV. The left-most column specifies how many agents were included, and down to what depth of nesting, in the recursive model structure.[6] As before, the more total threat removed indicates the better performance.

TABLE IV
PERFORMANCE AND COMPUTATION TIME OF THE AGENTS USING RANDOM, INDEPENDENT, AND ITERATIVE DEEPENING ALGORITHM IN THE *6 vs. 6* SETTING

Methods	Performance	Computation Time
Random	516.2 ± 126.6	0.013 ± 0.007
Independent	527.9 ± 175.3	0.017 ± 0.005
2 w/ depth 1	809.7 ± 105.1	0.040 ± 0.015
2 w/ depth 2	816.7 ± 108.9	0.056 ± 0.020
2 w/ depth 3	841.4 ± 82.1	0.058 ± 0.021
2 w/ depth 4	859.9 ± 57.9	0.062 ± 0.043
3 w/ depth 1	906.2 ± 98.3	0.364 ± 0.047
3 w/ depth 2	1001.3 ± 60.2	0.374 ± 0.053
3 w/ depth 3	1050.8 ± 46.3	0.381 ± 0.051
3 w/ depth 4	1055.0 ± 47.9	0.373 ± 0.054
4 w/ depth 1	1055.2 ± 48.9	3.036 ± 0.380
4 w/ depth 2	1055.4 ± 48.0	3.129 ± 0.501
4 w/ depth 3	1055.7 ± 50.8	3.168 ± 0.556
4 w/ depth 4	1055.9 ± 47.4	3.202 ± 0.570
5 w/ depth 1	1057.2 ± 48.0	32.533 ± 11.064
5 w/ depth 2	1057.4 ± 46.6	38.400 ± 7.026
5 w/ depth 3	1057.7 ± 46.3	37.675 ± 6.813
5 w/ depth 4	1058.1 ± 49.0	37.975 ± 7.108
6 w/ depth 1	1060.6 ± 45.3	144.776 ± 97.834
6 w/ depth 2	1060.8 ± 44.4	691.043 ± 166.245
6 w/ depth 3	1061.2 ± 44.3	689.943 ± 164.606
6 w/ depth 4	1061.5 ± 44.0	692.443 ± 164.804

The random agents showed the worst performance, 516.2, and spent the shortest computation time, 0.013. The performance (527.9) and the computation time (0.017) of the independent agents were similar to those of the random agents. The independent agent, maximizing its own expected utility without considering others, did not score well in this coordination experiment and obtained next to the worst performance. Compared with the performance of the independent agent, the performance of the agent[7] coordinating with one neighboring agent was 809.7. This is an increase by 53.4%. This case is a clear illustration that modeling other agents is advantageous for effective coordination.

As the depth or the breadth of modeling space increased, the performance improved as we expected. When the number of agents considered for coordination was two or three, the threat removed from our defense area increased from

[6]The Random agent could be labeled as 0 w/ depth 0, and Independent as 1 w/ depth 0, according to this method.

[7]In Table IV, this corresponds to 2 with depth 1.

809.7 to 859.9 when each defense agent considered the other agent, and increased from 906.2 to 1055.0 when it modeled two other agents. However, as more agents and further reasoning depth were included, the result of performance increase leveled off. The amount of improvement from 3 with depth 4 to 6 with depth 4 in Table IV was only $6.5 (= 1061.5 - 1055.0)$ due to more information.

As more breadth and more depth were considered, our agents also required more computation time. As more agents, between 2 and 5, were added, the running times range over $[0.040 - 0.062]$, $[0.364 - 0.381]$, $[3.036 - 3.202]$, and $[32.533 - 38.400]$, respectively. When all six agents were considered for coordination, it should be noted that the computation time at reasoning depth two (691.043) took five times as long as that of reasoning depth one (144.776). Proceeding to more than depth one, the agents (6 with depth 2 through 4) needed to compute additionally five six-dimensional payoff matrices and so used relatively more running time to construct and evaluate the matrices. Since the increase in dimensions of payoff matrices is equal to the increasing number of agents considered, the computation time spent depending on the number of agents was saliently different.

C. Determining the Scope of Modeling Using Value of Time

Our definition of urgency, expressed also in relation (8), gets at the intuition that an agent stands to lose utility as time progresses in an urgent situation if the agent is not doing anything to remedy the situation. For example, an air defense unit's not trying to intercept the incoming threats may cause the increase in the probability of increased damages. Thus, the situation is urgent in that the defense agent loses utility as time progresses.

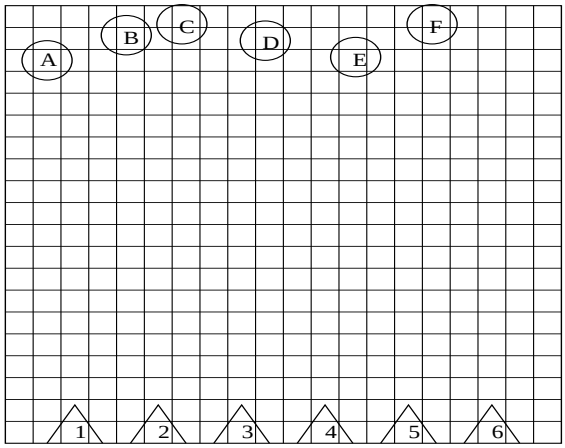
C.1 Example Scenario

As a simple example let us consider the scenario depicted in Fig. 6. This scenario has six incoming targets and six defending agents in a 20 by 20 grid world. Fig. 6 shows the initial positions of the defense agents and the incoming targets. As before, the defense agents are stationary, and the incoming targets move straight down at the rate of two grids for one second.

In this example, we focus on the agent 1's decision-making situation. Agent 1 randomly chooses one of the six targets for its default action which takes 0.013 second. Then agent 1 compares the benefit and the cost of computation, as described in relation (8). First, agent 1 computes the cost of computation given its randomly selected target, say D. At current time step the interception probability is

$$At\ time\ t, P(H_{1D}) = 0.979008.$$

After one second target D will be closer to the defense agents by two grids, the angle between the battery's line of sight and D increases from 21 to 23, and the interception probability from the agent 1's perspective decreases as follows:



Targets: A B C D E F
Positions: (2,3) (5,2) (7,1) (10,2) (13,3) (16,1)

Batteries: 1 2 3 4 5 6
Positions: (3,20) (6,20) (9,20) (12,20) (15,20) (18,2

Fig. 6. An example anti-air defense scenario. There are six defense units and six incoming targets.

$$At\ \ time\ \ t+1, P(H_{1D}) = 0.976680.$$

Given the warhead size of target D of 250, the interception probabilities at different time steps, and the computation time for deliberating longer (the decision-making using independent strategy requires 0.017 computation time, as described in Table IV), the cost of computation is given by[8]

$$EU(\alpha|E,t) - EU(\alpha|E,t+0.017)$$
$$= (0.979008 - 0.976680) \times 250 \times 0.017 = 0.009894.$$

Given the performance profiles in Table IV, the benefit of computation (the left hand side of relation (8)) is

$$EU(\alpha|E+e,t) - EU(\alpha|E,t) = 527.9 - 516.2 = 11.7.$$

Therefore, in this example scenario, since the benefit of computation is greater than the cost of computation, the agent should not execute the immediately available action of shooting at a randomly picked target D at the initial time t, but it should consider the extra information e (consisting of the agent's own decision-making situation), compute $EU(\alpha|E+e,t+t')$, and be ready to execute the best action at time $t+0.017$.

C.2 The Appropriate Scope of Modeling in 6 vs. 6 Settings

It turns out that the break-even point, at which the agent should cease computation, is to model 3 agents with depth 3 in the example scenario. Let's assume that the parameters of the particular situation at hand change. For example, if the targets move faster and get closer, or if the

[8]The urgency, or the value of time, can be computed as 0.582 per second.

computation is slower, the optimal scope of reasoning is smaller and less computation will take place. In general, the factors of the calculation of value of time include the capability of data processing machine and the external factors imparting urgency to the situation.

We tested the notion of urgency in the anti-air defense settings which contain six defense units and six incoming targets. To identify the appropriate scope of modeling given the offline pre-computed performance profiles (Table IV), we had each defense agent calculate the benefit and the cost of computation for a given scope of modeling in a sample of 100 runs. We found that the agents stopped reasoning when the average breadth of modeling space was 3 and the average depth of modeling space was 3.017. That is, they considered the adjacent two agents with the reasoning depth just over three, on the average. This resulted in coordinated decision-making of optimal quality, given the speed of the incoming targets and the agents' own computational capabilities.

VII. Conclusion

In time-critical settings, autonomous agents need to perform rational decision-making. Our work contributes to bounded rationality in multi-agent systems, when decision-making time is crucial. We formulated two robust procedures which combine reactivity and deliberation under time pressure. First one uses rules which summarize regularities of the domain and recommend which actions should not be deliberated over. The second procedure uses the definition of urgency, or the value of time, and a performance profile computed off-line. We derived a formula which combines on-line estimation of the value of time with information contained in the performance profiles to choose whether to act or to keep on computing.

We tested our agent's performance in scaled-up settings and found that the compiled rules were useful in reducing agents' deliberation time while maintaining reasonable performance levels. We also tested how our agents identify the proper scope of modeling information using the value of time. We have shown how the urgency can be computed during on-line processing and how autonomous agents can flexibly determine whether to act or to think. We believe we have moved a step closer to the goal of practical and flexible decisions in coordinated multi-agent settings. We provided two multi-agent decision-making procedures. Each is showing promising results in experiments in the high-stakes anti-air defense.

As part of our ongoing work, we are applying our flexible decision-making procedures to the predator-prey game [24]. The goal of the predator agents is to coordinate their actions to capture one prey agent by surrounding it on all four sides. Like the anti-air defense domain above, this domain also exhibits a trade-off between decision quality and computation time. While relying on reactive rules might degrade the quality of decision-making of a predator, relying on deliberative but delayed decisions might decrease the probability of capturing the prey as well. The behaviors of predator agents using various strategies to surround the prey while closing in on it, can be compiled into reactive rules from a set of scenarios. Further, as in the anti-air defense domain, the performance profiles can be used to quantify the quality of decision-making depending on the number of other predator agents considered and on the depth of reasoning. Based on performance profiles and the computed urgency of the situation, the predator agents can decide whether they should consider other agents or take the best current action without further deliberation. We hope to be able to develop more comprehensive agents through our future work on the above research issues, and to apply our framework to other time-critical domains.

Acknowledgments

The authors would like to thank the editors and the anonymous reviewers for their valuable comments and suggestions.

References

[1] H. A. Simon, "A behavioral model of rational choice," *Quarterly Journal of Economics*, vol. 69, pp. 99–118, 1955.

[2] I. J. Good, "Twenty-seven principles of rationality," in *Foundations of Statistical Inference*, V. P. Godambe and D. A. Sprott, Eds., pp. 108–141. Holt, Rinehart, and Winston, 1971.

[3] E. J. Horvitz, "Reasoning about beliefs and actions under computational resource constraints," in *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, 1987.

[4] E. J. Horvitz, G. F. Cooper, and D. E. Heckerman, "Reflection and action under scarce resources: Theoretical principles and theoretical study," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, Aug. 1989, pp. 1121–1127.

[5] T. Dean, "Decision-theoretic control of inference for time-critical applications," *Artificial Intelligence*, pp. 1–28, Nov. 1990.

[6] T. Dean, "Decision-theoretic planning and markov decision processes," Tech. Rep., Brown University, Computer Science Department, Providence, RI, 1994.

[7] S. J. Russell and E. H. Wefald, "Principles of metareasoning," *Artificial Intelligence*, vol. 49, pp. 361–395, 1991.

[8] S. J. Russell and D. Subramanian, "Provably bounded-optimal agents," *Journal of Artificial Intelligence Research*, vol. 2, pp. 575–609, 1995.

[9] S. Zilberstein and S. J. Russell, "Optimal composition of real-time systems," *Artificial Intelligence*, vol. 82, no. 1, pp. 181–213, 1996.

[10] I. Gilboa and D. Schmeidler, "Case-based knowledge and induction," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 30, no. 2, pp. 85–95, 2000.

[11] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Journal of Computational Intelligence*, vol. 4, pp. 349–355, 1988.

[12] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in *Proceedings of the 1st International Conference on Multiagent Systems*, July 1995, pp. 312–319.

[13] S. J. Russell, "Rationality and intelligence," *Artificial Intelligence*, vol. 94, pp. 57–77, 1997.

[14] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, Mar. 1986.

[15] P. E. Agre and D. Chapman, "Pengi: An implementation of a theory of activity," in *Proceedings of the National Conference on Artificial Intelligence*, Seattle, Washington, 1987, pp. 268–272.

[16] J. Fox and P. Krause, "Symbolic decision theory and autonomous systems," in *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, UCLA, California, July 1991, pp. 103–110.

[17] A. S. Rao and M. P. Georgeff, "An abstract architecture for rational agents," in *Proceedings of the Knowledge Representation and Reasoning*, 1992, pp. 439–449.

[18] R.A. Howard, "The foundations of decision analysis," *IEEE Transactions on Systems, Science, and Cybernetics*, vol. 4, pp. 211–219, 1968.

[19] E. J. Horvitz, "Reasoning under varying and uncertain resource constraints," in *Proceedings of the 7th AAAI*, Aug. 1988, pp. 111–116.

[20] M. Schut and M. Wooldridge, "Principles of intention reconsideration," in *Proceedings of the Fifth International Conference on Autonomous Agents*. 2001, pp. 340–347, ACM Press.

[21] V. R. Lesser, "Reflections on the nature of multi-agent coordination and its implications for an agent architecture," *Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 89–111, 1998.

[22] E. H. Durfee, "Practically coordinating," *AI Magazine*, vol. 20, no. 1, pp. 99–116, 1999.

[23] P. J. Gmytrasiewicz and E. H. Durfee, "A rigorous, operational formalization of recursive modeling," in *Proceedings of the First International Conference on Multi-Agent Systems*, 1995, pp. 125–132.

[24] P. J. Gmytrasiewicz, S. Noh, and T. Kellogg, "Bayesian update of recusive agent models," *User Modeling and User-Adapted Interaction: An International Journal*, vol. 8, no. 1/2, pp. 49–69, 1998.

[25] P. J. Gmytrasiewicz and E. H. Durfee, "Rational coordination in multi-agent environments," *Autonomous Agents and Multiagent Systems Journal*, vol. 3, pp. 319–350, 2000.

[26] G.E.G. Beroggi and W.A. Wallace, "The effect of reasoning logics on real-time decision making," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 27, no. 6, pp. 743–749, 1997.

[27] S. Noh and P. J. Gmytrasiewicz, "Multiagent coordination in antiair defense: A case study," in *Multi-Agent Rationality – MAAMAW'97 Workshop, Lecture Notes in Artificial Intelligence Vol. 1237*, pp. 4–16. Springer, New York, May 1997.

[28] S. Noh and P. J. Gmytrasiewicz, "Agent modeling in antiair defense," in *Proceedings of the Sixth International Conference on User Modeling*, Sardinia, Italy, June 1997, pp. 389–400, Springer-Verlag.

[29] S. Noh and P. J. Gmytrasiewicz, "Coordination and belief update in a distributed anti-air environment," in *Proceedings of the 31st Hawaii International Conference*, Hawaii, Jan 1998, vol. V, pp. 142–151, IEEE Computer Society.

[30] S. Noh and P. J. Gmytrasiewicz, "Rational communicative behavior in anti-air defense," in *Proceedings of the Third International Conference on Multi Agent Systems*, Paris, France, July 1998, pp. 214–221, IEEE Computer Society.

[31] R. H. M. Macfadzean, *Surface-Based Air Defense System Analysis*, Artech House, 1992.

[32] S. Noh and P. J. Gmytrasiewicz, "Uncertain knowledge representation and communicative behavior in coordinated defense," in *Issues in Agent Communication, LNAI 1916*, pp. 281–300. Springer, 2000.

[33] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley and Sons, New York, 1976.

[34] J. S. Rosenschein, "The role of knowledge in logic-based rational interactions," in *Proceedings of the Seventh Phoenix Conference on Computers and Communications*, Scottsdale, Arizona, Feb. 1988, pp. 497–504.

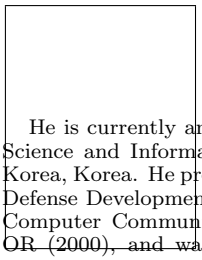
[35] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[36] E. H. Durfee and T. A. Montgomery, "MICE: A flexible testbed for intelligent coordination experiments," in *Proceedings of the 1989 Distributed AI Workshop*, Sept. 1989, pp. 25–40.

[37] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning Journal*, vol. 3, no. 4, pp. 261–283, 1989.

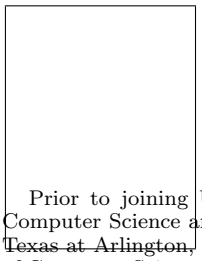
[38] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[39] R. M. Cameron-Jones and J. R. Quinlan, "Efficient top-down induction of logic programs," *SIGART Bulletin*, vol. 5, no. 1, pp. 33–42, Jan. 1994.



Sanguk Noh received a B.S. in biology, an M.S. in computer science and engineering from Sogang University, Seoul, Korea, and a Ph.D. in computer science and engineering from the University of Texas, Arlington, TX, in 1999.

He is currently an Assistant Professor in the School of Computer Science and Information Engineering at the Catholic University of Korea, Korea. He previously held research positions at the Agency for Defense Development, Korea (1989-1995), in the Center for Human-Computer Communication, Oregon Graduate Institute, Beaverton, OR (2000), and was an Assistant Professor in the Department of Computer Science at the University of Missouri, Rolla, MO (2000-2002). His research interests include decision theory, multi-agent systems, knowledge management, machine learning, and intelligent real-time systems.



Piotr J. Gmytrasiewicz is an associate professor in the Computer Science Department at the University of Illinois at Chicago. His research interests concentrate on rational artificial agents, distributed artificial intelligence, uncertainty reasoning, automated decision-making and on multi-agent coordination and intelligent communication.

Prior to joining UIC he has been an associate professor in the Computer Science and Engineering Department at the University of Texas at Arlington, a visiting Assistant Professor in the Department of Computer Science at the University of California at Riverside, and a researcher at the Artificial Intelligence Laboratory at the Hebrew University of Jerusalem. He received a Ph.D. degree from the University of Michigan in Ann Arbor in 1992.