Goal Oriented Extensive Games

Vittorio Amos Ziparo
Dipartimento di Informatica e Sistemistica “Antonio Ruberti”
Università di Roma “La Sapienza”
ziparo@dis.uniroma1.it
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
ziparo@informatik.uni-freiburg.de

ABSTRACT

In this paper we propose a framework for MultiAgent planning where heterogeneous self-interested agents, which inhabit a partially observable strategic environment, pursue possibly conflicting goals. We base our framework on Extensive Games: a Game Theoretic tool for analyzing the behavior of interacting agents. Compared to other approaches, we try to narrow down the sources of uncertainty to the actions performed by other agents. Moreover, communication is an integral part of the model and allows reasoning of distributed knowledge and action synchronization.

1. INTRODUCTION

In recent years increasing interest has been devoted to the study of organization of intelligent agents and the interactions among them. This is due to several motivations varying from the increasing development of networking infrastructures to the growth in the complexity of the problems to be addressed.

In this paper we focus on MultiAgent planning, which is the problem of finding a plan of action for a set of agents in order to reach a set of goals, given a specification of their initial state and actions.

In particular, we present a MultiAgent Planning framework for self-interested heterogeneous agents with different goals and different reward functions. For each agent i we define a set of goal states valued by a function w_i. The utility for reaching a state s is defined for each agent i as:

$$u_i(s) = v_i(s) - c_i(s) \tag{1}$$

where $c_i(s)$ is the cost for agent i of reaching s, and its valuation of the state s is defined as:

$$v_i(s) = \begin{cases} w_i(s) & \text{if } s \text{ is a goal state for agent } i. \\ -F & \text{if } s \text{ is a fail state.} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where $-F$ is a negative reward.

We may consider as a particular case the one where every agent has the same set of goals and the same valuation of their worth. Nevertheless, the utility for a state may still differ from agent to agent depending on the effort it spent to reach it.

We assume that agents inhabit a partially observable environment, because their sensors may not be able to give them access to the full description of the state at each point in time. Thus the description of the state is generally incomplete and the agents may need to gather information locally at execution time in order to achieve their goals.

From a global perspective, during execution, the current state can be reconstructed for the system with certainty but will be in general incomplete because some information may not have been gathered yet.

From a local perspective, the environment is deterministic except for actions of other agents. We refer to such an environment as strategic. In particular, we assume that agents are not able to perceive actions performed by the other agents.

Agents are thus uncertain, when reasoning about action selection, about which actions the other agents may have performed and about which information they have gathered.

To reduce the uncertainty, communication actions may be used. These actions have a cost and are explicitly represented in the model.

The remainder of this paper is structured as follows. We first give some examples in Section 2 to motivate our approach. Then we focus on strategic reasoning in Section 3, where we show a Game Theoretic model (and a related solution concept) called Extensive Game, which is capable of representing the relevant aspects addressed by the problem. Although this model is very powerful, it is infeasible to write it down by hand for realistic scenarios. For this reason, we provide in Section 4 a compact representation, using a Strips-like formalism and show how a particular subclass of the Extensive Games can be built from it. Finally, in Section 5 we present a discussion of the framework and a proposal for future work on the topic.

2. EXAMPLES

In this section we show two MultiAgent scenarios to motivate our approach. Let's first think of a set of agents embedded into PDAs, which have to organize a schedule for some users. The users will provide some goals such as to organize some meetings or go skiing.

The agents will have to coordinate in order to organize for the meetings but will also have to plan the activities for the users. An agent will prefer to organize the meetings in its user's office rather than incurring in the cost of having him to move from an office to an other. Moreover, a schedule where there is more free time for lunch or which reserves some time to leisure activities will be preferred to one which does not.

The agents will be self-interested in the sense that they want to maximize the utility of their user. Maximizing the utility of their own user may go against the interest of other agents (i.e. of their users). There will thus have to be some strategic reasoning which will have to take into account the use of communication in order to synchronize the activities and reveal some private information.

For example, one may want to condition their plans on some exogenous event he is aware of. This may be to set a meeting on one day depending if his flight is confirmed or if he finds a substitute for a lecture. On the other hand, the user may not want the agent to reveal all his private information or his schedule for privacy reasons. Thus reasoning about when and about what to communicate is necessary.

The second scenario we present is a Rescue scenario inspired by the competitions at RoboCup [4]. In a Rescue Environment, a group of heterogeneous agents seek victims in an unstructured environment (i.e a disaster scenario like a destructed building after an earthquake). In order to identify a victim correctly, robots have to provide three types of evidence out of six possible ones.

The robots have different sensors to prove evidences (e.g. recognize human form or sound signals) and different actuator capabilities (e.g climbing stairs or opening doors). This suits well to real scenarios where it is very hard to build robots which have both many sensing capabilities and a good mobility.

Thus cooperation and coordination will be necessary because agents, on the one hand, cannot achieve their goals alone, on the other hand, may conflict with one another while taking actions.

This is a cooperative setting where agents have the same goals and agree on their valuation. On the other hand, we assume that robots are provided by different organizations, which may be willing to rescue as many victims as possible but may not want their robot to take all the risks for doing this. In fact, robots may get damaged or destroyed during the rescuing operations in such a dangerous environment. This will result in a competitive situation, where agents try to minimize their risks given that the rescuing goal must be achieved.

From now on, for the simplicity of the resulting games, we will refer in our examples to the Rescue scenario.

3. EXTENSIVE GAMES

In this framework we are interested in studying the interaction of self-interested decision makers which are *rational* (i.e. pursue well defined exogenous objectives) and *reason strategically* (i.e. take into account their knowledge or expectations of other decision-makers' behavior).

This topic has been studied successfully in the Game Theoretic literature [7] which provides a set of analytical tools for understanding these phenomena. Most of these tools consist in a *model* of the problem, usually called game, and a set of *solution concepts* which can be applied to the game.

Thus, the first step for building our framework consists in finding an appropriate class of models capable of representing the relevant aspects which we would like to address and that were discussed in Section 1. We have chosen Finite Extensive Games with Imperfect Information for this purpose. In the remainder of this section we will give a brief overview of this tool, based on [7].

An Extensive Game is a detailed description of the sequential structure of the decision problems encountered by the players in a strategic situation. In these games, agents are not informed about the moves the other players will perform. Furthermore, players may be imperfectly informed about some of the choices that have already been made and about the other players' private information.

Before introducing the formal definition of Extensive Game, we will provide some notation which will be used throughout this paper: We define a profile a $x = (x_i)_{i \in N}$ as a collection of variables for each player. For any profile $x = (x_j)_{j \in N}$ and any $i \in N$, we let x_{-i} be the list $(x_i)_{i \in N \setminus \{i\}}$ of elements of the profile x for all players except i. Finally, a sequence of profiles will be defined as x^k.

We are now ready to define an Extensive Game [7]:

Definition 1. An Extensive Game has the following components.

- A finite set N (the set of players).
- A set H of sequences (finite or infinite) that satisfies the following three properties.
  - The empty sequence $\emptyset$ is a member of H.
  - If $(a^k)_{k=1,\ldots,K} \in H$ (where K may be infinite) and $L < K$ then $(a^k)_{k=1,\ldots,L} \in H$.
  - If an infinite sequence $(a^k)_{k=1}^{\infty}$ satisfies $(a^k)_{k=1,\ldots,L} \in H$ for every positive integer L then $(a^k)_{k=1}^{\infty} \in H$.

Each member of H is a history; each component of a history is an action taken by a player. A history $(a^k)_{k=1}^K \in H$ is terminal if it is infinite or if there is no (a^{K+1}) such that $(a^k)_{k=1,\dots,K+1} \in H$. The set of actions available after the nonterminal history h is denoted by $A(h) = \{a : (h,a) \in H\}$ and the set of terminal histories is denoted Z.

- A function P that assigns to each nonterminal history (each member of $H \backslash Z$) a member of $N \cup c$. P is the player function, $P(h)$ being the player who takes an action after history h. If $P(h) = c$ then chance determines the action taken after the history h.
- A function f_c that associates after every history h for which $P(h) = c$ a probability measure $f_c(\cdot \mid h)$ on $A(h)$, where each such probability measure is independent of every other such measure. $f_c(a \mid h)$ is the probability that a occurs after the history h.
- For each player $i \in N$ a partition $\mathcal{I}_i$ of $h \in H : P(h) = i$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the the player $P(h)$ for any $h \in I_i$. $\mathcal{I}_i$ is the information partition of player i; a set $I_i \in \mathcal{I}_i$ is an information set of player i.
- For each player $i \in N$ a preference relation $\succeq_i$ on lotteries over Z (the preference relation of player i) that can be represented as the expected value of a payoff function defined on Z.

We will now show a simple example from the Rescue domain introduced in Section 2. Moreover, in this example a tree representation for Extensive Games, called game tree, will be presented and used to build the model.

Example 1. Lets consider two heterogeneous robots which have to verify if there is a victim behind an open door, thus sense two potential evidences (e1 and e2) for it. The first robot can close the door and sense evidence e1. The second one can only sense e2 and needs the victim to be visible in order to correctly perform the action.

We can build the game as a tree as shown in Fig. 1. The color of non-terminal nodes of this tree represents the player who moves at that turn. We will use red for player one, blue for player two, and green for chance (nature). The mapping from nodes to players that we define for this game is the player function.

At the root node, given our implementation of the player function, player one moves. He may choose between closing the door and sensing property e1. In the latter case, the next move will be performed by chance, which will determine with equal probability if the property is true or false.

If the property e1 is false, the game will end because there is no victim. Each agent will be awarded with an utility. In this case, agent one has a negative utility because of the cost of performing the action.

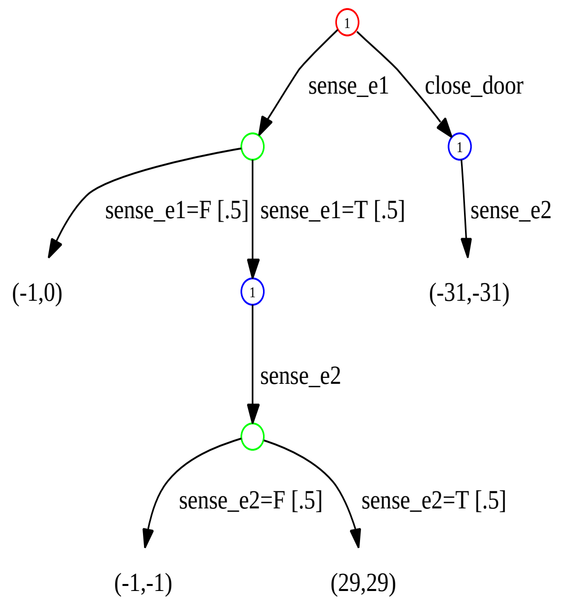

Figure 1: The game tree of a simple game from the rescue domain

Since agent two cannot perceive the actions of the other agent, from its viewpoint, the two nodes at which it is its turn to play are indistinguishable and thus are in the same information set. This is represented by the number one in the node, which is a unique identifier of the information set for the player. The constraint that the agent must be able to perform the same actions at all the nodes in the same information set is satisfied. In fact, it will sense e2 in both cases.

It is interesting to notice that, if agent two tries to sense e2 after the door has been closed, it will fail and both agents will get an utility which is the result of a negative reward (for the failure) plus the cost of performing the actions. In the other case, after its move, nature plays and determines if the agents reached the goal (i.e. if they effectively identified a victim).

Once the structure of the game has been defined, we must specify what the possible plans are which agents may use as a basis for finding a solution. These plans are called strategies in Game Theory and are formally defined as [7]:

Definition 2. A *pure strategy* of player $i \in N$ in an Extensive Game $\langle N, H, P, f_c, (\mathcal{I}_i), (\succeq_i)\rangle$ is a function that assigns an action in $A(I_i)$ to each information set $I_i \in \mathcal{I}_i$.

A player may also randomize over the actions in a strategy or over pure strategies [7]:

Definition 3. A *mixed strategy* of player $i \in N$ in an Extensive Game $\langle N, H, P, f_c, (\mathcal{I}_i), (\succeq_i)\rangle$ is a probability measure over the set of player i's pure strategies. A *behavioral strategy* of player i is a collection $\beta_i(I_i)_{I_i \in \mathcal{I}_i}$ of independent probability measures, where $\beta_i(I_i)$ is a probability measure over $A(I_i)$.

Thus agents must agree on set of strategies; But the question remains which ones. Although many solution concepts in the Game Theoretic literature exist for Extensive Games with Imperfect Information, in this paper we will focus on Nash Equilibria. The other solution concepts are subsets of Nash equilibria and aim to rule out those equilibria which are not credible. In this paper we will follow a different approach: We will select the equilibria which maximize the social welfare (i.e. the product of utilities) and among those the ones which maximize the system performance (i.e. the sum of utilities) [8].

Thus we can now define a Nash equilibrium for an Extensive Game [7]:

Definition 4. A *Nash equilibrium* in mixed strategies of an Extensive Game is a profile $\sigma*$ of mixed strategies with the property that for every player $i \in N$ we have:

$O(\sigma^*_{-i}, \sigma^*_i) \succeq_i O(\sigma^*_{-i}, \sigma_i)$ for every mixed strategy σ_i of player i.

where O is a function returning the expected value of the utility for each agent, when all commit to a profile $\sigma*$ of mixed strategies.

4. A COMPACT REPRESENTATION

Game trees which model non-trivial problems are usually huge, and it is practically infeasible to build them by hand. It is thus necessary to find a way to represent them more concisely.

In this section we present a novel approach for representing and building a subset of the Extensive Games which we call Goal Oriented Extensive Games. These will be characterized by having conflict-free situations. These property is enforced while building the tree and may be acheived with the use of communication. In fact, we introduce in our representation communication primitives and their operational semantics, which allow for reasoning of distributed knowledge and action synchronization.

We represent the game in terms of: i) a description of the initial state, ii) a description of the goals, iii) the timing constraints of the agents, iv) a description of the actions, and v) the number of agents. In particular, actions are described in Strips-like fashion, with some minor extensions for non-instantaneous actions and for dealing with heterogeneous agents. Furthermore, we add the possibility for an action to sense a property in the environment. The description of an action has the following components:

- a unique *action name.*
- a *set of agents* who may perform the action.
- *preconditions* which must hold before the action can be executed.
- *during conditions* which must hold throughout the action execution.
- *effects* which must hold after the action execution.
- if it is a sensing action, the *sensed property.*
- a function $t : S \times N \rightarrow \Re^+$ returning the time needed for agent $i \in N$ to perform the action in a given state $s \in S$.
- a function $c : S \times N \rightarrow \Re$ returning the cost for agent $i \in N$ to perform the action in a given state $s \in S$.

We build the tree associated with the game by expanding the reachable states for a given time horizon. This consists of a directed graph whose edges represent actions. Furthermore, a node has the following properties:

1. For each agent i, an *information set* is_i consisting of all the actions that the agent i knows to have occurred.
2. The *history* $h \in H$ at the current node.
3. The *player turn.*
4. Current *timestamp* for each agent.
5. If the node is terminal, an *utility* u_i for each agent i.

We first initialize the history of the root node and its information sets with the empty set. The first player will be player one, and the timestamps will be set to zero. We then expand the reachable nodes in a breadth first search.

For each node that we want to expand we apply all the actions which are applicable at the belief state of the agent in turn to play and which can be executed within the time constraints. The belief state and the global state can be respectively reconstructed, applying the actions of the information set of the current player and of the history to the initial state.

The successor node can be built from the current node applying an action in the following way: at first, we set the history h and the information set is_i of the current player i, by updating them from the current node with the performed action. We then set the next player for the successor node with the function $P(h)$.

In order to implement the player function P, an ordering between agents defines the next player at the successor node. For example, if we have two players, after player one, player two will play and then player one will play again. If the performed action is a sensing action, the player for the successor node will be nature, while the player that would have played at that node will play after nature moves. If there are no applicable actions at a node, the player gives up his turn and the next player in the ordering will play.

Nature (i.e. chance) is a special player that always plays the same behavioral strategy whatever the other players do. In this framework, we assume that nature may perform two actions: One revealing that the sensed property is true and one that it is false. In the following examples, for simplicity, we will assume that $f_c(a \mid h) = 0.5 \;\; \forall a \in A(h)$, but different functions can be implemented depending on the domain. The player who performed the sensing action will be able to observe nature's effective moves, and his information set will be updated accordingly.

Finally, we update the timestamp at the successor node for the player who performed the action at the current node. This is trivially the current timestamp plus the estimated time to perform the action.

We allow agents to continue acting also after having achieved some goal until this satisfies time constraint. This is because goal nodes with a higher utility may eventually be reached.

Every node which has no applicable action for any player, and thus cannot be further expanded, will be a terminal node. The utility values for these nodes are calculated using formula 1.

The proposed framework is dealing with goal oriented agents. Thus we would like to rule out all those solutions (in this case Nash Equilibria) where no agent reaches a goal. To bias this behavior, we collapse the subtrees that do not reach any goal to single leaf nodes of the game tree.

It may be noticed that the constraint that $A(h) = A(h')$, whenever h and h' are in the same information set, is satisfied. In fact, for each information set, we will have the same belief state and timestamp for each agent, thus the same actions will be applicable at all its nodes.

4.1 Interactions Among Actions

To safely execute actions, we need to understand interactions among them. In order to execute two actions in parallel, since we have no control over their rates, all the interleavings must be possible. Taking into account the limited effects of actions [1], we can define the actions as *commuting* if each of preconditions, during conditions and effects of each action is satisfiable with all the conditions that define the other action.

Even though two actions do not commute, we can still execute them safely by suspending one action during the execution of the other. For this to be possible, the preconditions of the suspended action have to be satisfied after the termination of the other action.

Let us assume that, for every history, actions performed by each agent commute with the actions performed by the other agents. In this case, the strategies are causally independent. We can enforce this assumption by not expanding nodes that correspond to histories where there is at least an action of an agent which does not commute with at least an action of another agent. These nodes will be labeled as failure nodes, and a penalty will be assigned to every agent.

On the other hand, from a strategical viewpoint, if each agent, as we assumed, cannot perceive the actions performed by the others, the situtation in which the players are involved is essentially the same as the one captured by a game in which they choose actions simultaneously.

It is worth noticing that, although the structure of the tree is sequential, during execution agents act asynchronously.

4.2 Communication

If the domain is tightly coupled, the commuting action assumption may be overly constraining. In fact, two non commuting actions may sometimes safely be executed if some ordering constraints are added. To order actions between agents we need some synchronization primitive which can be implemented through communication.

Let us define the set of agents which can communicate as ca and ca_i as $ca - \{i\}$. We will add for each agent $i \in ca$ the set of actions $communicateto(r)$ for all $r \in ca_i$ and a *donothing* action. If an agent i performs the latter action, it gives up its future turns to play until it receives a communication from an agent j. When an agent i receives a communication from agent j, his timestamp for that node is updated with $max(timestamp(s, i), timestamp(s, j))$.

We can now expand branches of non commuting actions if they are safe when ordered and a communication enforces such ordering. With the use of communication, we may now have ordering constraints between actions performed in a local plan with actions from the other agents' plans.

Communication may be useful, not only to suspend activities during regions of the plan, but also to enrich the local knowledge of agents. As previously mentioned, agents may not be informed about the actions the other agents have performed. Moreover, the information about nature moves is distributed upon the information sets of agents who performed sensing actions. Thus, if all the local information sets were merged, the global (incomplete) state would be known with certainty.

The only relevant information for the strategic reasoning in an Extensive Game are the actions which have been performed. We can thus use the known performed actions as the information to be exchanged. In particular, we can communicate any subset of the information set. By adding communication, we allow local information sets to contain actions relative to other players.

We can thus, through the use of communication, achieve action synchronization when necessary and have the opportunity to reconstruct locally larger portions of the global state.

In order to avoid long sequences of communications, we constrain the applicability of this type of actions. In fact, we communicate if for at least one node of the actual information set this is informative for the recipient. To be informative means that in the sender's information set there is at least an action which is not in the recipient's information set.

As the following example shows, communication is not only needed for informing about sensed information or action synchronization, but it allows more tight cooperation when necessary. In the example, for the sake of simplicity, we will restrict our attention to communications which inform about the complete information set.

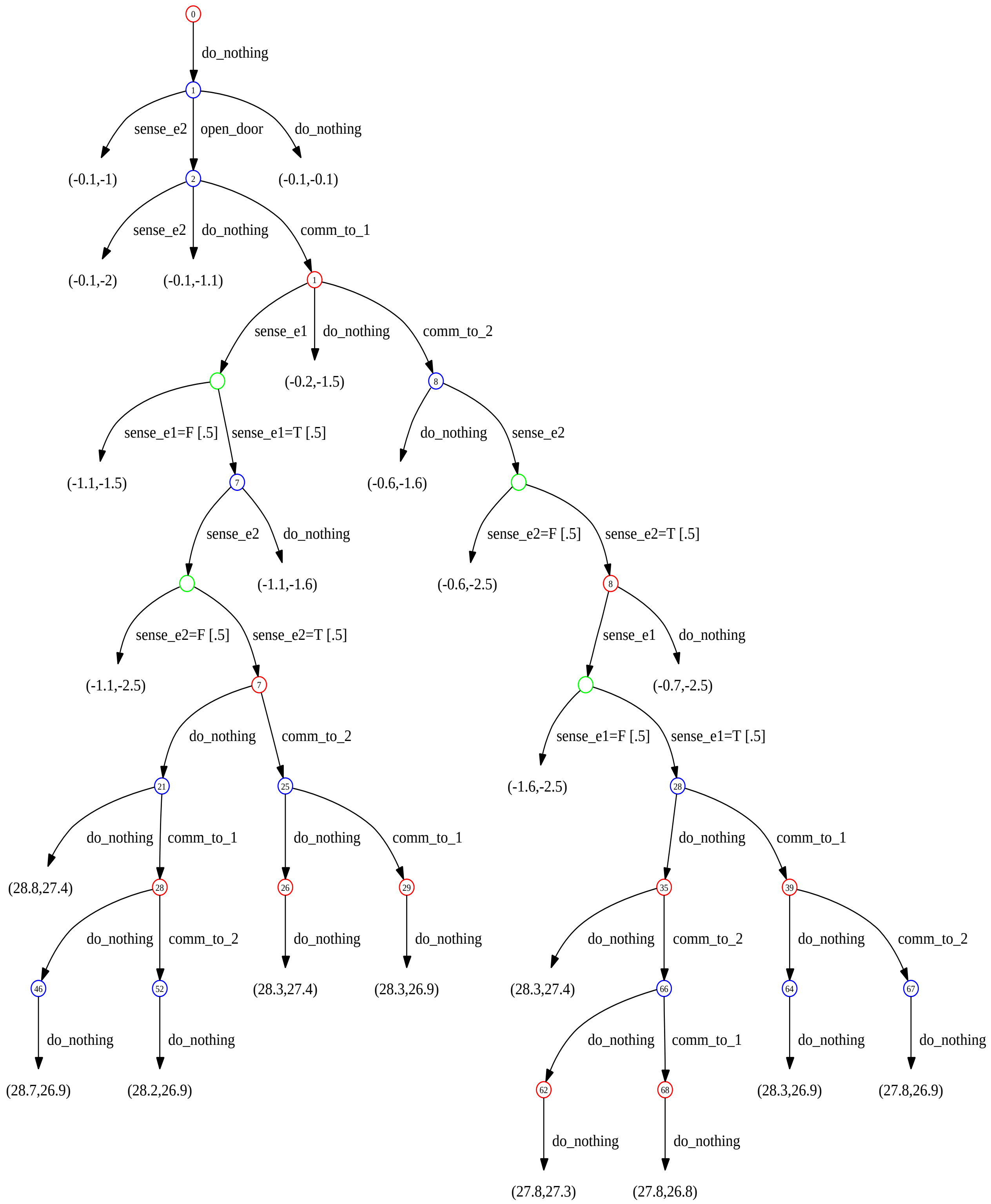

Figure 2: A game tree for the rescue domain using communication.

4.3 Example

Consider two heterogeneous robots in the Rescue domain. The goal is to bring the evidence of a victim at a location which is behind a door. If the victim is insight, one agent can use a camera to look for human form (evidence $e1$), while the other can open a door and can sense the temperature of objects behind the door (evidence $e2$). In the initial state agents only know that the door is closed.

The time needed to perform each action is $500ms$ with the cost of one. *donothing* has a duration that depends on the reception of a communication, as previously described, and has the cost of 0.1. Communication has the cost of 0.5 and is instantaneous (but may not be in general).

The goal states are those where both $e1$ and $e2$ are true and the agents have one second to reach one of such states.

Without communication, our algorithm would produce an empty tree because, in this case, it is impossible to reach a goal and thus not worth playing any game. In fact, if we analyze the game more deeply, since the applicability of actions is based only on local knowledge, the first agent will never know that the door has been opened and thus never sense for the victim.

Fig. 2 shows the game tree produced by our algorithm. It can be noticed that, due to the timing constraints, the only possible plan starts with agent one waiting for agent two opening the door and communicating the accomplishment of the action.

Although a strategy for a player prescribes an action for every information set (even though some information sets will never be reached if every agent follows the strategies in the equilibrium), we will define a plan as the subsets of the strategies in the equilibrium which refer to reachable histories.

Let us have a look at the plan resulting from one of the Nash equilibria: agent one does nothing, while agent two opens the door and communicates. At this point, player one senses the property $e1$ because is informed that the door has been opened. If the property is false, the game ends (there is no way of achieving the goal); if it is true, player two will sense $e2$. In the latter case, if the result of the sensing action is true, both players will do nothing because there is no way to further improve the worth of the goal.

5. DISCUSSION AND FUTURE WORK

In this paper we propose a framework for MultiAgent planning where heterogeneous agents are self-interested and pursue potentially conflicting goals. We base our framework on Extensive Games, a Game Theoretic tool for analyzing the behavior of interacting agents.

In the MultiAgent planning literature there are several approaches which deal with self-interested agents. In particular POSGs [3] are a framework which extend Dec-POMDPs [2] for having different reward functions for each agent. These models are used to represent partially observable stochastic games, where actions have stochastic effects and the environment is partially observable.

It has been shown in [5] that a finite horizon POSG can be modeled as an Extensive Game. From a computational perspective, the POSGs are very hard to solve, and our approach is to try to narrow down the sources of uncertainty to simplify the problem.

In fact, in our framework, we assume that the only source of change in the environment are the agents in the system and that their actions are deterministic. Thus, we constrain our reasoning to predicting their rational behavior. Moreover, we mitigate the sources of uncertainty, relative to the performed actions and private information gathered, through the use of communication actions which are defined as an integral part of the model.

We implemented the proposed algorithm and fed the results to gambit [6], which is a library of Game Theory software and tools for the construction and analysis of finite extensive and strategic games. The preliminary results show that, although the games we produce can represent all the relevant aspects of the class of problems we want to address, finding a solution is infeasible for all but trivial games. Thus, we need to develop a more efficient algorithm for building and solving goal oriented extensive games.

In particular, our future work involves studying the complexity of the problem and designing an efficient algorithm for solving non trivial cases. This could be achieved by building a reduced game tree (trying to exploit the structure of the problem domain) and by finding Nash Equilibria with some approximation technique.

6. REFERENCES

[1] M. P. Georgeff. Communication and interaction in multi-agent planning. In A. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 200–204. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[2] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res. (JAIR)*, 22:143–174, 2004.

[3] E. Hansen, D. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games, 2004.

[4] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Conf. on Man, Systems, and Cybernetics(SMC-99)*, 1999.

[5] H. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games II*, pages 193–216, 1953.

[6] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory, version 0.2005.12.12, 2005.

[7] M. J. Osborne and A. Rubinstein. *A course in game theory.* MIT Press, 1994.

[8] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers.* MIT Press, Cambridge, Massachusetts, 1994.