Probabilistic Goal Recognition - Grid World

Francesco Chiariello

January 4, 2019

Contents

1 Introduction

Goal recognition is the task of inferring the intents and goals of an agent based on its observed actions. As such, it requires a model that allows to represent the world in which and on which the agent operates. This model, then, allows to perform an high-level reasoning on the agent's decision-making process. The ability to understand this process, i.e., how and why an agent takes its decision, is an indispensable skill for every cognitive social agent, enabling it, moreover, to predict the behaviour of the other agents.

The ability to recognize goals and plans is, in fact, a fundamental cognitive capability in humans as well and is a prerequisite for communication, since it presuppose an ability to understand the motivations of the participants and subjects of the discussion. If we aim to build intelligent machines able to interact in a natural way with humans as well as with other machines it is therefore clear that they must be equipped with the ability to understand others' goals. Such a machine would be able to predict the needs of other agents and proactively act in order to help achieving that objectives.

It is not surpring that goal recognition is useful in domains like Human Robot Interaction (HRI) and multi-agent system (MAS) for applications like intelligent personal assistants, intelligent tutoring systems, natural language understanding, smart environments, and team coordination.

Several different approaches have been applied to the goal recognition problem, including: hand-coded action taxonomies, probabilistic belief networks, consistency graphs, Bayesian inference, and AI planning. In the AI planning approach a planning system is used to determine how closely a sequence of observed actions matches plans for each possible goal. For each goal, this is done by comparing the cost of a plan for that goal with the cost of a plan for that goal that includes the observed actions.

In this project we will see an implementation of this approach for a Grid World [Fig 1]. Here user controls an actor (the robot) moving in a grid world, heading it toward one of the goal cells (in red) present in the grid. While moving, system estimates where actor is headed computing and updating a probability distribution over the set of goal cells. Since a probability $\mathbb{P}(G|O)$ of the goals given the observed actions is produced, the problem is also referred to as **probabilistic** goal recognition.

The rest of the paper is organized as follows. In section 2 are recalled basic notions and definitions of planning. Section 3 defines formally the probabilist goal recognition problem and explain the technique used to solve it. Section 4 describes the assumptions and limitations underlying this technique and its implementation. Section 5 shows and comment one run of the application. Finally, Section 6 summarize the work done and discuss possible extensions.

2 Planning Background

AI planning is the computational study of the problem of choosing and organizing actions in order to satisfy some goal specifications. One of the first and most famous formalism for that is the STRIPS.

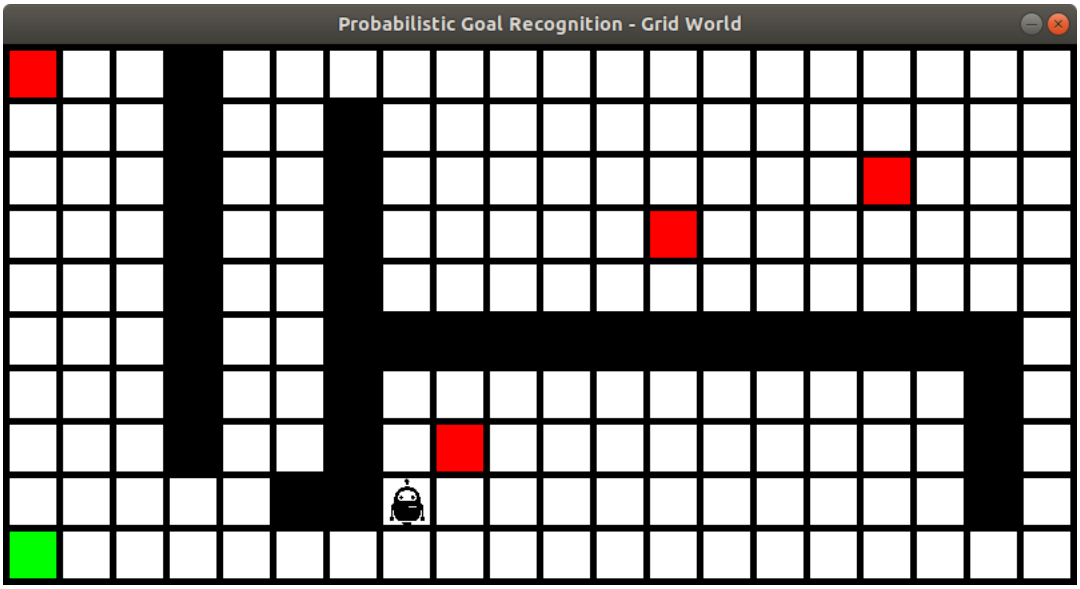

Figure 1: A screenshot of the grid world.

STRIPS. A STRIPS planning problem is a tuple $P =< F, I, A, G >$, where F is a finite set of propositional state variables (also called *fluents*), $I \subseteq F$ and $G \subseteq F$ are the initial and goal situations, and $A \subseteq \mathcal{P}^3(F)$ is the set of actions $a =< Pre(a), Add(a), Del(a) >$.
A problem P defines a state model. A state s is a set of fluents and is a compact way to represent interpretations over F: fluents in s are assumed to be true, the others to be false. In this model the initial state is $s_0 = I$ and the goal states s_g are the one such that $G \subseteq s_g$.
An action a is *applicable* in a state s if $Pre(a) \subseteq s$, and, if so, applying a in s result in a new state $s' = (s \setminus Del(a)) \cup Add(a)$.
A solution plan for P is an applicable action sequence that maps the initial state into a goal state. Each action $a \in A$ has an associate non-negative cost and the cost of the plan is the sum of the costs of the actions that compose it. A solution plan is *optimal* if it has minimum cost.

PDDL. Planning Domain Definition Language is a standard encoding language for *classical* planning problems. It was developed by Drew McDermott and his colleagues in 1998 for the First International Planning Competition. The adoption of a common formalism for describing planning domains allowed more direct comparison of systems and algorithms.
PDDL allows to state planning tasks, such as STRIPS problems. Planning tasks specified in PDDL are separated into two files: a domain file and a problem file.
In the domain file are specified predicates, i. e. properties of objects, and actions. In the problem file are specified the objects of interest, the initial state and the goal specifications. The domain-problem pair can then be given as input to a classical planner that, hopefully, returns a solution plan or a failure in case no solution exists.

3 Probabilistic Goal Recognition

A probabilistic goal recognition problem is a tuple $T =< P, \mathcal{G}, O, \mathbb{P} >$, where $P =< F, I, A >$ is a planning domain, $\mathcal{G}$ is a set of possible goal states G, O is the observed sequence of actions, and $\mathbb{P}$ is the prior probability distribution over $\mathcal{G}$.
The solution to the probabilistic goal recognition problem is the posterior probability $\mathbb{P}(G|O)$ over $\mathcal{G}$, given the observed actions O. From the Bayes rule,

$$\mathbb{P}(G|O) = \alpha\, \mathbb{P}(O|G)\, \mathbb{P}(G)$$

where α is a normalizing costant.

The challenge in this formulation is the definition of the likelihoods $\mathbb{P}(G|O)$, expressing the probability of observing O when the goal is G. A rationality postulate is that G is a better predictor of O when the difference $\Delta(G, O) = c(G, O) - c(G)$ between the cost $c(G, O)$ of an optimal plan for G complyiant with O, and the cost $c(G)$ of an optimal plan for G, is smaller. The likelihoods can then be characterized in terms of a Boltzmann distribution

$$\mathbb{P}(O|G) = \frac{e^{-\beta\Delta(G,O)}}{1 + e^{-\beta\Delta(G,O)}}$$

where β is a positive constant.

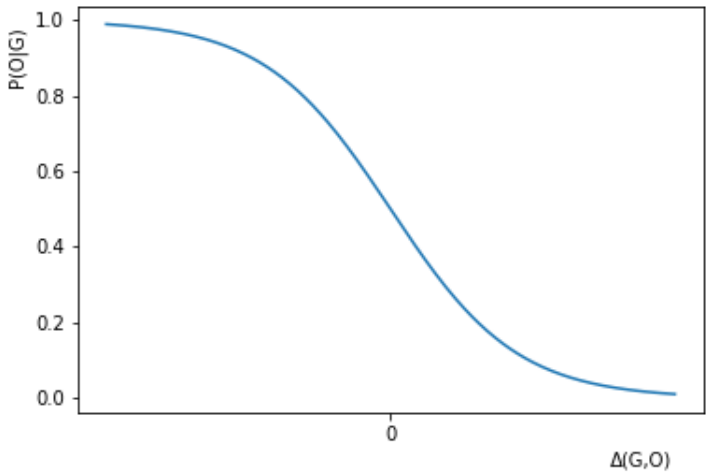

Figure 2: Plot of the Boltzmann distribution.

4 Assumptions and Implementation Details

The approach as described can be applied to a variety of different problems, as long as they match the (albeit restrictive) assumptions underlying it. It is worth make them explicit. First of all, the actions must have deterministic effects and both the actor and the observer

must have complete information about the initial state. Second, the observed actions must be correct. In other words, the approach is not able to handle uncertainty in observations. For this reasons, it is hard to apply it in real-world applications. It is however not required to the observer to know *all* the sequence of actions, allowing gaps in it. Therefore, a plan is compliant with the observed actions if all the actions of the sequence appear in the plan and the order in which they appear in the sequence is respected, i.e., if observed actions form a subsequence (and not necessary a substring) of the plan.

The other set of assumptions is about the actor behaviour and capabilities. Using a probability distribution over the set of goals in fact, presume that the actor is trying to reach one and only one of the goals. This can be incorrect for a variety of different reasons: (i) the actor is trying to reach a goal that is not modeled, (ii) the actor is not trying to reach a goal at all, (iii) the actor is trying to reach multiple goals, like in temporally extended goals.
Further it is assumed that the actor is able to compute the optimal (or at least near-optimal) plan to the goal it wants to pursue and that it is following that plan.
These are all real issues in the developed project since the actor is controlled by a user. *It is assumed that the user will try to reach goal cells with the minimun number of actions,* but this can not be the case. Maybe the user is just wandering the world without a goal, perhaps just to understand how the control works. Maybe the user just tend to use horizontal and vertical commands more than diagonal ones, altough this could be attenuated giving to a diagonal move a greater cost.

In the project is assumed that the observer, i.e., the application itself, has access to the whole sequence of actions. The cost $c(G, O)$ then is the cost of the optimal plan for reach G from the state resulting from O plus the cost of O. This cost must be computed for each goal, each time an action is executed. Costs $c(G)$ are instead computed only once. All this costs are computed using the classical planning system Fast Downward [3] with A* search and the landmark-cut heuristic, which is admissible [4]. Note that the PDDL domain does not try to model the grid world as appears to the user but it is functional to computing costs. Note also that the presence of obstacles make this computation hard and requiring in fact a planner, instead of using, for example, a domain-dependent technique like coordinate differences.

5 Results

In this section are shown and discuss results on one example [Fig 3]; users are encouraged to test the application on the configurations they like.
Here the actor move straight from the bottom left cell to the up left cell. Probabilities are in [Fig 4 - 5]. We can see that at the start all the goals have the same probability: this is the default prior probability $\mathbb{P}(G)$. After the first action a probability update is executed. System realizes that the first goal (remember goals are ordered from bottom-up and from left to right) is not reachable, assigning to it a zero probality: this is handled manually since the approach described above does not consider failures in the planning. The third goal sees a progressive increase in the cost to reach it and so a decrease in its probability. Consequently the second goal sees an increase in its probability. The temporary increase in the probability assigned to the third goal after the first action is due to the diagonal move and to the decrease of the probability of first goal.

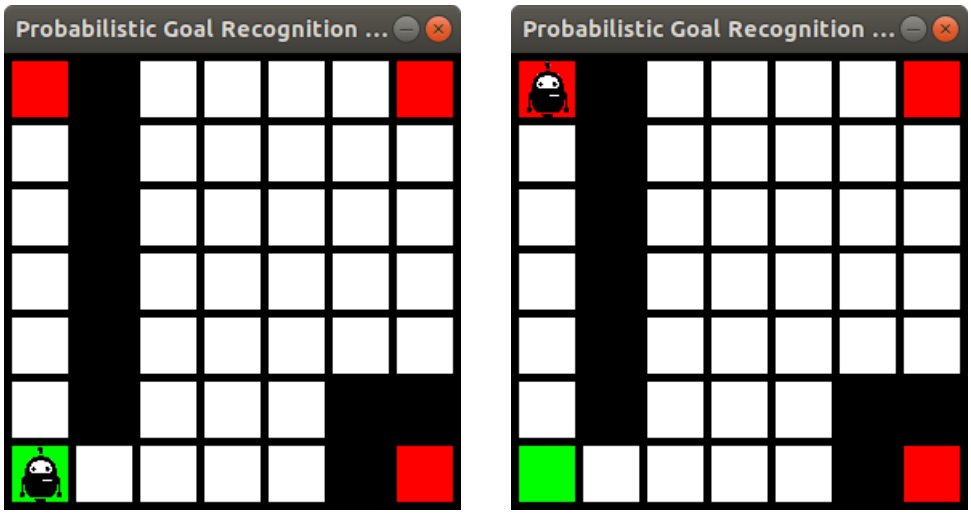

Figure 3: The actor move up passing from the configuration on the left to the one on the right.

6 Conclusions and Future Work

For this project I implemented a probabilistic goal recognition technique for estimating the goal of an actor moving in a grid world. It can be seen, playing with the application, that the probability updates are mainly consistent with the possible expectations. The only things may seem odd are due to diagonal moves and the fact that they allow in effect to make two moves at the cost of one. To solve that problem could be assigned to diagonal moves a cost greater than the default 1 and, obviously, smaller than 2, in order for that moves to be still relevant. For example could be assigned to them a cost of $\sqrt{2}$, which has a nice geometric meaning.
Finally, possible future works could be the extension of the implementation to allow incomplete observed action sequences, or the investigation of techniques to handle noisy observations allowing the application of them to real-world problems.

References

[1] Miquel Ramirez, Hector Geffner. *Probabilistic Plan Recognition using off-the-shelf Classical Planners*, 2010.

[2] Yolanda E-Martın, Marıa D. R-Moreno, David E. Smith. *A Fast Goal Recognition Technique Based on Interaction Estimates*, 2015.

[3] Malte Helmert, *The Fast Downward Planning System*, 2006.

[4] Malte Helmert, Carmel Domshlak. *Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?*, 2009.

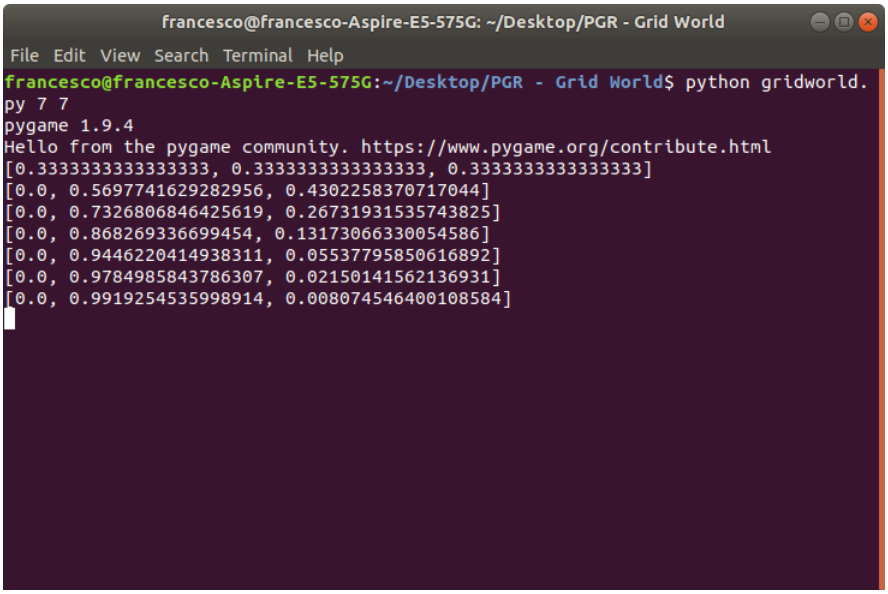

Figure 4: Probabilities printed in terminal.

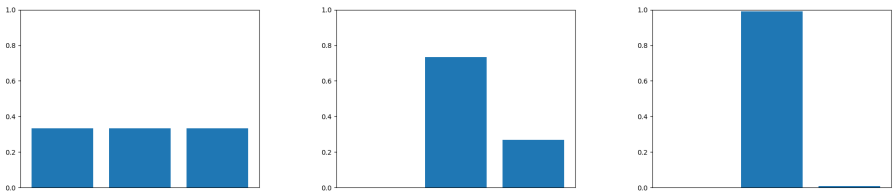

Figure 5: Probabilities bar graphs in *plots* folder. Here are reported the initial state, an intermediate state and the final state.