An Interaction-Based Model for Situated Agents

Bruno Bouzy
Jeune Equipe InfoCom - UFR de Mathématiques et d'Informatique
Université René Descartes (Paris 5)
45, rue des Saints-Pères 75270 Paris cedex 06 FRANCE
tel: 33 (0)1 44 55 35 58 - fax: 33 (0)1 44 55 35 35
e-mail: bouzy@math-info.univ-paris5.fr

Abstract

This paper enriches the current studies on interaction with a domain where agents have original properties about the interaction. The agents interact with the external world and the agent-world interactions enable the model to recognise the approximate state of the agents. They interact between each other and the agent-agent interactions refine their state. This model has been implemented with success in Indigo, a Go playing program.

1. Introduction

Recent researches in Artificial Intelligence use principled characterizations of interactions between agents and their environment to guide the analysis of living agents and the design of artificial ones [1]. In order to highlight the interaction principle, this paper compares the result of a multiagent system in which the agents interact with the environment only, with a system where the agents interact with the environment and also between each other. This model is applied to a complex domain where the agents are situated : the game of Go. The model is implemented and used in the Go playing program Indigo.

2. Ontology of a position

An *agent* is a *connected* set of intersections of the same colour. The basic property of an agent is to be *situated* on a set of intersections. The intersections that belong to the morphological closure [2] of an agent, without belonging to the agent nor to the adverse agents, constitute the *inside* of the agent. The intersections that belong to the dilate set of an agent, without belonging to the agent nor to its inside nor to the adverse agents, make up the *outside* of the agent. An agent may be *alive* when its inside is sufficiently big. In this case, an agent is spatially autonomous because its state does not depend on its outside. An agent may also be *weak* when its inside and its outside are both sufficiently small. In order to determine the state of a weak agent, the sole use of the inside and the outside of the agent is not enough and the use of *interactions* becomes necessary.

3. The interactions

There are two kinds of inter-agent interactions : the *friendly* interaction and the *adverse* interaction. Two neighbouring agents that may connect themselves in one move, have a friendly interaction. An agent has a *friendly state* that reflects the number of its friendly interactions. We define the 3-uple T(agent) = f(I, E, A) for each agent where

I is its inside, E its outside and A its friendly state. The domain's theory gives a partial order that allows a two by two comparison between adverse agents. An *adverse interaction* between two agents reflects the result of this comparison. The number of friendly or adverse interactions is not limited. Each agent has an *adverse state* that represents the states of the adverse interactions. When a weak agent has adverse interactions that are all negative, its adverse state is also negative and the agent is said to be *dead*. When an agent dies, the adverse agents group themselves with it and make up a unique new agent. The adverse agents are like predators that are eating the prey agent. The set of intersections on which the new agent is situated is the union of the intersections on which the adverse agents are situated. While agents are eating dead ones, the situation and the set of the agents change and the properties of agent are recomputed. When no change is detected, the building of the description stops.

4. Results and discussion

Indigo is a Go playing program that uses, among other tools, the interaction-based model that is presented here. A conceptual description of the whole program can be found in [3]. An eight page description can be found in [4]. Indigo's manpower is about three man-years. It contains 35 000 lines of C++. It is ranked on the Computer Go Ladder [5].

During a game, for each position, Indigo builds the agents, their interactions and their states in order to build the evaluation function. This evaluation function is complex. It leads to a high computational cost. Therefore, it is not used by a tree search and the move decision process in Indigo is simple : Indigo chooses the move that is attached to the biggest agent whose state is not alive. If there is no such agent (in calm positions), Indigo plays the move that fills the biggest empty space. This move decision process is of course simple. Its main weakness is missing big moves on calm positions (because it may focus on small instable agents that do not have strategical importance).

When a move is made, the position may completely change at the agent level because an agent that was present before the move may have several outcomes after the move : modified, cut, connected or killed... Therefore the overhead of maintaining the agents is very high. This is 95% of the computational activity of Indigo. (Other 5% are devoted to the move decision process).

The performance of Indigo actually depends on many other factors not discussed here and of course this is extremely difficult to substantiate our claim about interaction with the sole Indigo's level. Inside the distributed AI community, this work shows once again the relevance of the interaction concept. It is an illustration of interaction and agency with several - about 50 - situated agents interacting in a relatively complex domain. Our contribution is a classification of the properties and relationships of the situated agents : the *inside*, the *outside*, the *friendly interactions*, the *adverse interactions* and the *stability*. We believe that these properties are sufficiently general and can be implemented in most of the multiagent systems in the future.

References

[1] Agre P.E., Computational research on interaction and agency, *Artificial Intelligence* 72 (1), pp. 1-52, 1995.

[2] Serra J., *Image Analysis and Mathematical Morphology*, Academic Press, 1982.

[3] B. Bouzy, Modélisation cognitive du joueur de Go, Ph.D. of Paris 6 University, France, 1995.

[4] B. Bouzy, The Indigo Program, *Proceedings of the Second Game Programming Workshop in Japan*, pp. 191-200, Hakone, 1995.

[5] Petersen E., The Computer Go Ladder, http://cgl.ucsf.edu/go/ladder.html