General Game Playing

General Game Playing consists in building programs to play the games they have never met before. A Game Master sends the rules to the participants and after a delay devoted to game analysis, the programs start playing. It can be applied to games with any number of players, with alternative or simultaneous moves, for zero-sum or collaborative game types.

General Game Playing avoids the shortcomings of current specialized game playing programs that cannot adapt to other domains than the games they were programmed for. At the very least it induces a broad exploration of the characteristics of decision situation to identify automatically the heuristics that may give good results in this situation.

The Game Description Language

The Game Description Language (GDL) is used to describe a game. It is based on first order logic, hence missing arithmetic. The following figure contains a representation in GDL of a binary version of the simultaneous play game My father has more money than yours [Berlekamp, Conway, & Guy[1982]. Keywords of GDL are written in upper case.

```
(ROLE left) (ROLE right)
(LEGAL (DOES tplayer (tell 0)))
(LEGAL (DOES tplayer (tell 1)))
(= (NEXT (value ?p ?x))
(= (value ?p (tell 7))
(= (value ?p (tell 7x))
(= (DISTINCT ?x ?y))
(= (GOAL ?p 0) (TRUE (value ?p 0))
(= (other ?x ?y) (role ?x) (role ?y))
(= (PROLOG) (ROLE right))

The rules indicate that there are two players (left and right), enumerates the legal moves (telling a figure), identify the terminal nodes (after the first and only move) and the reward for each player: 0 for the smaller figure, 100 for the greatest and 50 in case of tie. The description of this game does not need the INIT keyword, used to describe the initial state of the board.

In the following, we designate the current situation of the game as the board status, even for games that are not played on a board.

Ary uses Prolog for rule interpretation

We made the choice to use a Prolog Interpreter as an inference engine for the interpretation of the rules of the game.

Monte-Carlo explorations

The Monte-Carlo implementation is straightforward: until the expiration of the thinking time, the current board status is loaded into the interpreter, legal moves are generated, and a random game is played until arriving in a terminal board status. The score is asked to the interpreter and accumulated in a counter associated with each of the first legal moves.

When the thinking time expires, the current game is stopped and the move with the best mean for Ary is chosen. Ignoring the scores of the other players has a clear advantage in simplicity: it is not necessary to distinguish between games that have one, two or more players; zero-sum games and cooperative games are treated identically. Moreover, we expected it to provide more interesting play, and it was in agreement with our goal to have Ary plays its best moves. Finally, the round-robin nature of the qualifying phase made it uninteresting to try to limit the score of the opponent.

UCT tree construction

Ary uses Upper Confidence bounds applied to Trees, usually called UCT. UCT adds to Monte-Carlo explorations of the games move tree an informed way to choose the branches that will be explored. A subset of the move tree is constructed incrementally, with a new node added for each Monte-Carlo exploration. On the next exploration, a path is chosen in the already built move tree by choosing the branch whose gain is maximum, as estimated by the Monte-Carlo algorithm plus confidence in the estimation, calculated by a function of the number of explorations of the node and of the number of exploration of the branch as \(\sqrt{\log t/d}\), where \(t\) and \(d\) are the number of leaves and the value of node. When arriving at a leaf node of the move tree, if it is not a terminal situation (i.e. it is not a leaf of the abstract move tree), then a new node is added to the tree and a Monte-Carlo simulation is started to obtain an evaluation of this node, also used to update the evaluation of the parent nodes [Kocsis & Szepesvári[2006]].

References


Ary: A Program for General Game Playing

Jean Méhat, Tristan Cazenave
Université de Paris 8, Saint Denis, 93536 Cedex, France
email: {jm,cazenave}@ai.univ-paris8.fr