General Game Playing

Mohit Agarwal, Ujjwal K. Singh

April 17, 2013

Mentor : Amitabha Mukerjee

Abstract

A General Game Playing agent plays game without any previous knowledge of the game and without any human intervention. A formal description of game, GDL(Game Descriptive Language) is provided to the player before the start of the game. This work implements the recent work [4] published by developer of CadiaPlayer which involves the major two changes in the MCTS(Monte Carlo Tree Search).

1 Introduction

General Game Playing Agents are AI Programs that tackles the problem of playing more than one game efficiently. They are very different from Specialized Game Players ex. DeepBlue. They do not rely on advanced specific algorithms designed to play a specific game. In specialized game players, much of the AI and design work is done by programmer only and there left a much less work for program to perform, Instead General Game Player analyzes the rules and regulations of the game and play accordingly as to maximize the score of the player.

General Game Players have no idea of game before the start i.e. they play those games that are not previously encountered by them. They do not require any human intervention. Thus General Game players are independent intelligent agents that accept descriptions of arbitrary games at runtime and perform their best by using such descriptions.

In General Game Playing, we mainly focus on games having finite state machines.

2 Motivation

Any real life situation can be carved into a digital game. If a General Game Player is able to tackle or play that digital game, it will be able to tackle that real life situation also. They have applications in Droids, UAVs, army defense and Robots (ex. Roomba Vacuum Cleaning Robot).

Researchers have been taking interest in this field as they provide testbed for AI of how computational procedures achieve intelligent behaviour. They add excitement and allow one to compare his/her skills to those of others. More intelligent agents are highly likely to win, so they can also be used in evaluation techniques for intelligent systems.

3 GDL (Game Descriptive Language)

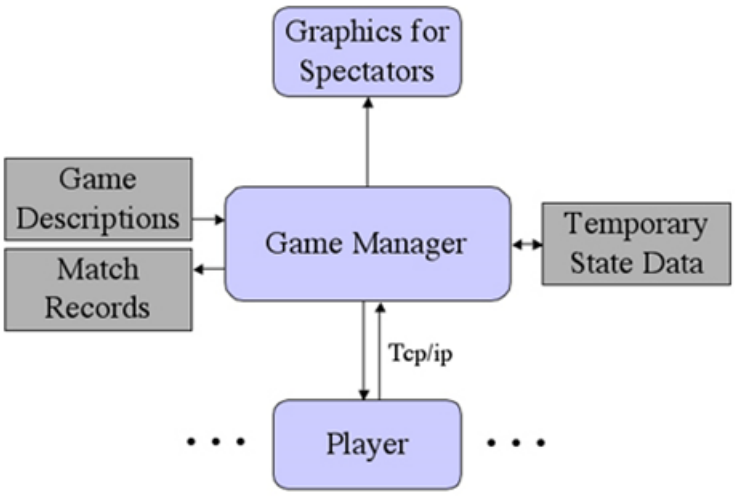
General Game Players require a formal description of rules and regulations of the game at runtime. As we concentrate on games having finite state machines, it is possible to represent these games in the form of state graphs. GDL encodes such games in more compact form as compared to direct representation.

It is logic Programming Language whose semantics are purely declarative which conceptualizes the game in terms of entities, actions, propositions and players. Entities are represented by object constants for ex. X as X player, cell (1,1,b) as cell at (1,1) marked with X in Tic-Tac-Toe. Actions are taken by players ex. Noop for no-operation-performed, mark(m,n) to mark cell (m,n) in Tic-Tac-Toe. Propositions are logical statements that are either true or false in a particular state of game. Some words are kept reserved in GDL that can be used for all games while meaning of other words are game dependent.

GDL can not describe games having element of chance or some sort of randomness. Hence, Improved version of GDL was released known as GDL-II [5] which can describe non-deterministic games ex. Poker etc.

```
(ROLE XPLAYER) ; to define player X
(ROLE OPLAYER)  ; to define player O
(INIT (CELL 1 1 B)) ;initial state
(INIT (CELL 1 2 B))
(INIT (CELL 1 3 B))
(INIT (CELL 2 1 B))
(INIT (CELL 2 2 B))
(INIT (CELL 2 3 B))
(INIT (CELL 3 1 B))
(INIT (CELL 3 2 B))
(INIT (CELL 3 3 B))
(INIT (CONTROL XPLAYER)) ; X player turn to make move initially
(<= (NEXT (CELL ?M ?N X)) (DOES XPLAYER (MARK ?M ?N)) (TRUE (CELL ?M ?N B)))
; if (m,n) is blank and X marks it, next state is (m,n,X)
(<= (NEXT (CELL ?M ?N O)) (DOES OPLAYER (MARK ?M ?N)) (TRUE (CELL ?M ?N B)))
; if (m,n) is blank and O marks it, next state is (m,n,O)
```

4 GameMaster



[logic.stanford.edu]

As the whole world will be busy in designing more and more intelligent gaming agents, a need of platform arises where they can compete against them and compare intelligence of their players. GameMaster provides the platform which communicates with players through TCP/IP Protocol, provides them GDL, their role in game (ex. X or O in Tic-Tac-Toe), startclock and payclock. Startclock is the time given to player before the start of game to analyze GDL. Payclock is the time given to player to think before making move. As players communicate with GameMaster their move, GameMaster stores Temporary state data of the game. It also maintains the match and tournament records.

Using Temporary State Data, GameMaster simulates the whole game in graphics for spectators.

5 Previous Work

Stanford Logic Group Initiatives and introduction of General Game Playing competition in AAAI Conference from 2005 promoted research in this new emerging field. Since 2005, tremendous research activity has been noted in the field. Winners of GGP Competition are:

Year	Player	Writer	University
2005	ClunePlayer	Jim Clune	UCLA
2006	FluxPlayer	Stephan Schiffel, Michael Thielscher	Dresden University
2007	CadiaPlayer	Yngvi Bjrnsson, Hilmar Finnsson	Reykjavik University
2008	CadiaPlayer	Yngvi Bjrnsson, Hilmar Finnsson	Reykjavik University
2009	Ary	Jean Mhat	Paris 8 Univeristy
2010	Ary	Jean Mhat	Paris 8 Univeristy
2011	TurboTurtle	Sam Schreiber	
2012	CadiaPlayer	Hilmar Finnsson, Yngvi Bjrnsson	Reykjavik University

Table 1: GGP Competition Winners

[source : wikipedia]

5.1 ClunePlayer

ClunePlayer utilize the concept of constructing heuristic evaluation functions [2] from GDL. These heuristic evaluation functions represent exact values of the simplified games. Simplified games are abstract models of the original games that incorporate payoff, control and termination. Exact values calculated are approximated for similar types of games. Thus the main drawback with cluneplayer is that it performs well only for similar types of game.

5.2 CadiaPlayer

Enhanced MCTS(Monte Carlo Tree Search) in the context of GGP. Two improvements in Basic MCTS has been done in which one allows early termination and second one improves action-selection strategy when both explored and unexplored actions are available. In addition, use of extensive game theory enhances its performance.

6 Algorithm

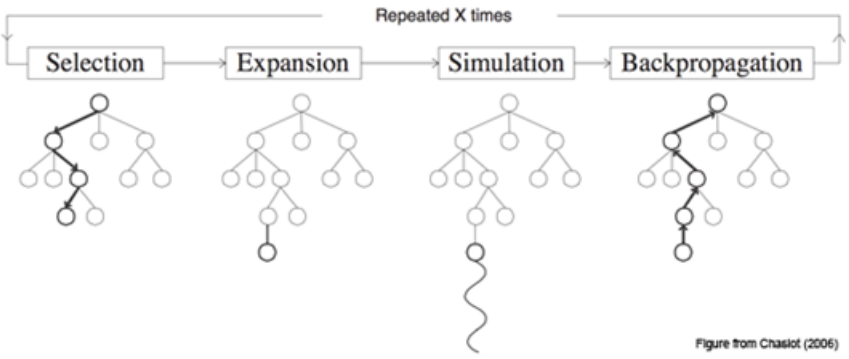
There are algorithm such as Alpha Beta and MiniMax which are efficient only when good evaluating function can be computed efficiently. They are good for games which have not large branching factor, say within 35.

Basic General Game Player [from Marius Schneider, Potsdam 2011].
It uses Monte-Carlo search which is another algorithm used by GGP for optimizing their best move. It select the best move to play by playing the large number of random games from the current state to the end and calculate the score for them.

Moves are evaluated according to the average score of the random games which it has played, not only at the beginning but at every stage of the game, using the bandit approach. **Bandit approach** is one in which we are expected get maximum reward.

It obviously has a problem as average score of a branch sometimes does not always provide the best move to be played thus spending more time in bad branches.

6.1 Monte Carlo Tree Search Basic Algorithm



[3]

1. Obtain an initial game tree in the time provided by the Game Manager as Start Clock.
2. Repeat the following sequence till timeout occurs or best move has been evaluated
   (a) Selection : From the current node/state as root, pick one path to a leaf with the best score" using a mini-max formula.
   (b) Expansion : From the best leaf, expand it by one level by creating a new child node which can be reached from parent by selection one legal move.
   (c) Simulation : Now the game simulated to get the score for the game by selecting random legal moves till the terminal node is reached. [1]
   (d) Back Propogation : Now retrace the path up till the root node and updating the score associated with the nodes which comes in the path.
3. Pick the best move of the root as your move when the timeout occurs.

Here it can be seen that:

- Some moves are bad and do not need further exploring.
- Should spend some time to verify whether a move that is current good will remain good or not.
- Need to have a mechanism for moves that are bad because of extremely bad luck to have a chance to be reconsidered later.

6.2 UCB: Upper Confidence Bound

For each node N_i, compute its UCB_i

$$UCB_i = \frac{W_i}{N_i} + C\sqrt{\frac{N}{N_i(a)}} \tag{1}$$

Where,
W_i is the reward sum associated with the node.
N_i is the total number of times it has been visited..
N is the total number of games played..
$N_i(a)$ is the times action a has been taken..
C is a constant called exploration parameter..

Expand a new simulated game for the move with the highest UCB value.
Constant C is used to keep the balance between.
Exploitation: exploring the best move so far.
Exploration: exploring other moves to see if they can be proved to be better.

6.3 Monte Carlo Tree Search Basic Algorithm with UCB

1. Obtain an initial game tree in the time provided by the Game Manager as Start Clock.
2. Repeat the following sequence till timeout occurs or best move has been evaluated
   (a) Selection : From the current node/state as root, pick one path to a leaf with the best score **May decide to trust the score of a node if it is visited more than the threshold. May decide to prune the node if its score is too bad now to save time.**
   (b) Expansion : From the best leaf, expand it by one level by creating a new child node which can be reached from parent by selection one legal move. **May decide to expand only the best leaf, or some potentially good leaves.**
   (c) Simulation : Now the game simulated to get the score for the game by selecting random legal moves till the terminal node is reached.
   (d) Back Propogation : Now retrace the path up till the root node and updating the score associated with the nodes which comes in the path.
3. Pick the best move of the root as your move when the timeout occurs.

7 Our Work

In early phase of the project we have studied basic lectures on general game playing from We have built our work on **Basic General Game Player [from Marius Schneider, Potsdam 2011]** which has implemented the Simple Monte Carlo using the **Bandit approach** which selects the best move depending upon the max score of the nodes.

We have extended and integrated Monte Carlo Tree Search (MCTS) to the Basic Game Player which is another search algorithm and used Upper Confidence Bound (UCB) to select the move for the player. We have used reference from [http://mcts.ai/index.html] for MCTS and UCB with exploring urgency from our main reference [4].

Then main idea for these extension is to explore more new nodes in early phase of the game playing. As in early phase large part of the game tree remains unexplored so the gaming should be designed so that it explores in early phase. In the later phase of game playing priority should be given to the nodes that has already been explored because it is just the reverse situation from the early phase as most of the tree has already been explored, thus exploitation should be given priority over exploration.

8 Results and Conclusion

Note: We have used Start Clock to be 10 sec for every experiment.

Game(Single Player)	Random	Legal	Minimax	Basic Player	Our Player
Buttons	Runtime: 32ms Score: Lose	Runtime: 39ms Score: Lose	Runtime: 195ms Score: Lose	Runtime: 32466ms Score: Win	Runtime: 32465ms Score: Win
Maze	10 steps Lose	10 steps Lose	10 steps Lose	7 steps Win	7 steps Win
Snake	6ms Score(100): 6	19ms 15	265ms 15	63779ms 42	106726ms 100
Peg	215ms 20	215ms 0	51645ms 0	71529ms 48	106674ms 100
memory small	81ms 25	72ms 0	344ms 0	24640ms 75	24632ms 100
blocks world	3ms 0	2ms 0	200ms 0	20726ms 100	20723ms 100

Figure 1: Score and runtime for various player for single player games Play Clock is 5 s

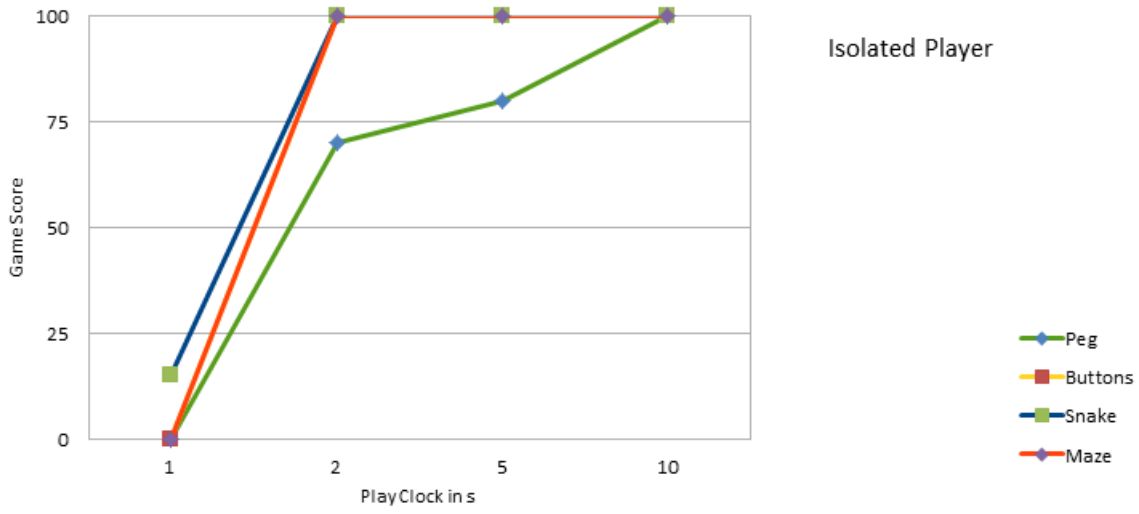


Figure 2: Score for our Game Player against Play Clock Time for various single player games

Game	Vs. Random	Vs. Basic Player
Tic-tac-toe	Results : 100 0 runtime (in ms):103609 23	results: 100/0 runtime (in ms): 3637 2 36376
		results: 50/50 runtime (in ms): 44243 44242
Checkers	results: 100 40 runtime (in ms):709219 771	results: 80/100 runtime (in ms): 393917 392234
		100/90 337430ms 337234ms
Mini Chess	Results: 100 0 runtime (in ms):8573 1 23	results: 100/0 runtime (in ms): 20755 20748
		results: 100/0 runtime (in ms): 20755 20751
Conn4	Results : 100/0 runtime (in ms): 95051 317	results: 100/0 runtime (in ms): 98990 99002
		results: 0 100 runtime (in ms): 118648 118692

Figure 3: Score of our player and other Players in dual playing games Play Clock is 5 s.

Play Clock Time	Tic-tac-toe	Checkers	Mini chess
1 sec	100/0	100/70	100/0
2 sec	100/0	100/80	100/0
5 sec	50/50	100/90	100/0
10 sec	50/50	100/80	100/0

Figure 4: Competition mode: Our Player vs. (Basic Player using Bandit Approach)

Introduction of Confidence bound on Monte Carlo Search Tree have significantly improve the time game agent takes to get an good estimate on best move to be played in the current situation. From the above table it can be seen that if the time provided is less than it plays very efficiently against the player which uses Bandit approach to select the move. UCB factor improves the time taken to search for optimized best move

Score is given in the scale of 0 to 100 (0 means lose and 100 means win or it is score in case of snake and checkers). 100/0 means that our player scored 100 and basic player scored 0.

Here Start Clock is the time provided to the Gaming Agent to learn the rules of the game through Game Descriptive Language and Play Clock is the time between the moves.

References

[1] Yngvi Bjornsson and Hilmar Finnsson. "a simulation-based approach to general game playing,. *IEEE Transactions on Computational Intelligence and AI in Games,*, 1, 2009.

[2] James Clune. "heuristic evaluation functions for general game playing,". *AAAI Conference,*, pages 1134–1139, 2007.

[3] Finnsson and Yngvi Bjornsson. "game-tree properties and mcts performance,. *The IJCAI-11 Workshop on General Game Playing,*, 2011.

[4] Hilmar Finnsson. Generalized monte-carlo tree search extensions for general game playing,. *The Twenty-Sixth AAAI Conference on Artificial Intelligence,*, pages 1550–1556, 2012.

[5] Michael Thielscher. "a general game description language for incomplete information games,. *AAAI Conference,*, 2010.