General Game Playing

Tomáš Musil

http://tomasm.cz/

04. 12. 2013

Motivation

"A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. ***Specialization is for insects.****"*

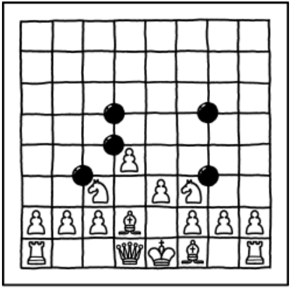
—Robert A. Heinlein, Time Enough for Love, 1973

"In general, we are building tools that amplify the human ability."

—Steve Jobs, 1980

Contents

1. Introduction
2. Game representation
3. Metagaming
4. GGP with ants
5. References



WHITE TO CONTINUE INSISTING THIS IS A CHESSBOARD

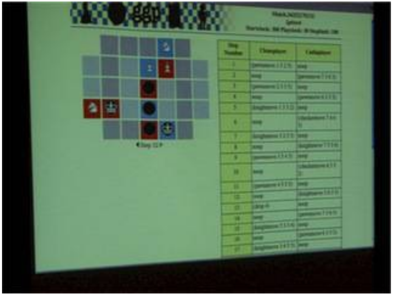
Introduction

General Game Playing

- aims at developing game playing agents that are able to play a variety of games
- raises questions about the nature of inteligence
- testbed for artificial inteligence
- develops new techiques for
  - descriptive programming
  - planning and scheduling
  - reasoning and formal verification
  - expert systems
  - machine learning
  - knowledge representation
- Jacques Pitrat: Realization of a general game playing program (1968)

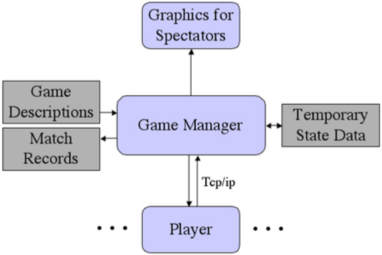
The International General Game Playing Competition

- since 2005
- organised by Stanford Logic Group of Stanford University
- held annualy at AAAI Conference



Game play

- Game Manager
- GGP Protocol
  - (info)
  - (start *id role description startclock playclock*)
  - (play *id move*)
  - (stop *id move*)
  - (abort *id*)
- Game Description Language

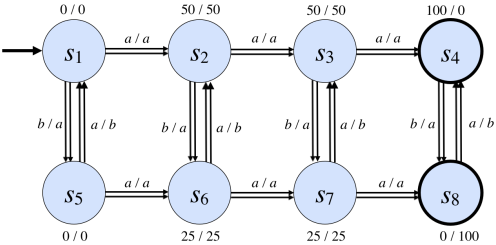


Game representation

- games must be
  - finite
  - synchronous (all players move on every step)
  - with complete information
  - playable
  - weakly winnable

Game Description Language

- logic programming language, variant of Datalog
- similar to Prolog (or Lisp - prefix GDL), except:
  - purely declarative
  - restrictions:
    - safety
    - stratified recursion and negation
    - no nested functional terms
  - every term can be computed in finite time
- structured state machine



Game Description Language: rules

role(a) means that *a* is a role in the game.

base(p) means that *p* is a base proposition in the game.

input(r,a) means that *a* is an action for role *r*.

init(p) means that the proposition *p* is true in the initial state.

true(p) means that the proposition *p* is true in the current state.

does(r,a) means that player *r* performs action *a* in the current state.

next(p) means that the proposition *p* is true in the next state.

legal(r,a) means it is legal for role *r* to play action *a* in the current state.

goal(r,n) means that player the current state has utility *n* for player *r*.

terminal means that the current state is a terminal state.

Game representation example: Tic Tac Toe

```
role(white)
role(black)

base(cell(M,N,x)) :- index(M) & index(N)
base(cell(M,N,o)) :- index(M) & index(N)
base(cell(M,N,b)) :- index(M) & index(N)

base(control(white))
base(control(black))

input(R,mark(M,N)) :- role(R) & index(M) & index(N)
input(R, noop) :- role(R)

index(1)
index(2)
index(3)
```

Game representation example: Tic Tac Toe

```
init(cell(1,1,b))
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(white))
```

```
legal(W,mark(X,Y)) :-
    true(cell(X,Y,b)) &
    true(control(W))

legal(white,noop) :-
    true(control(black))

legal(black,noop) :-
    true(control(white))
```

Game representation example: Tic Tac Toe

```
next(cell(M,N,x)) :-
    does(white,mark(M,N)) & true(cell(M,N,b))

next(cell(M,N,o)) :-
    does(black,mark(M,N)) & true(cell(M,N,b))

next(cell(M,N,W)) :-
    true(cell(M,N,W)) & distinct(W,b)

next(cell(M,N,b)) :-
    does(W,mark(J,K)) & true(cell(M,N,W)) &
    distinct(M,J)

next(cell(M,N,b)) :-
    does(W,mark(J,K)) & true(cell(M,N,W)) &
    distinct(N,K)

next(control(white)) :- true(control(black))
next(control(black)) :- true(control(white))
```

Game representation example: Tic Tac Toe

```
goal(white,100) :- line(x) & ~line(o)
goal(white,50) :- ~line(x) & ~line(o)
goal(white,0) :- ~line(x) & line(o)

goal(black,100) :- ~line(x) & line(o)
goal(black,50) :- ~line(x) & ~line(o)
goal(black,0) :- line(x) & ~line(o)

line(X) :- row(M,X)
line(X) :- column(M,X)
line(X) :- diagonal(X)
```

Game representation example: Tic Tac Toe

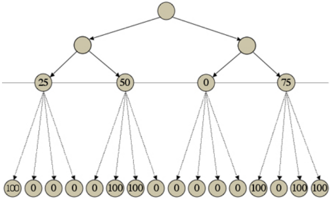
```
row(M,X) :-
    true(cell(M,1,X)) & true(cell(M,2,X)) &
    true(cell(M,3,X))

column(M,X) :-
    true(cell(1,N,X)) & true(cell(2,N,X)) &
    true(cell(3,N,X))

diagonal(X) :-
    true(cell(1,1,X)) & true(cell(2,2,X)) &
    true(cell(3,3,X))

diagonal(X) :-
    true(cell(1,3,X)) & true(cell(2,2,X)) &
    true(cell(3,1,X))
```

Game representation example: Tic Tac Toe

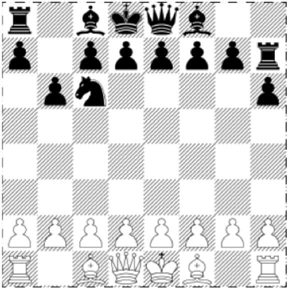
```
terminal :- line(W)
terminal :- ~open

open :- true(cell(M,N,b))
```

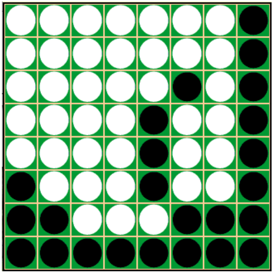
Game play

- expanding the game tree
- min-max
- Monte Carlo
  - winning the AAAI competition since 2007
- general heuristics
  - mobility
  - focus
  - goal proximity
  - combination
- metagaming

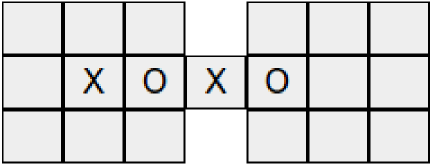


Hodgepodge





Joint Tic Tac Toe



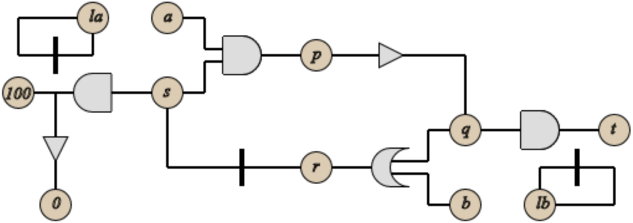
Metagaming

- analysis of propositional nets and rule graphs
  - compact representation
- proofs using logic
- finding structure in games:
  - factoring
  - bottlenecks (Triathalon)
  - symmetry detection (Tic-Tac-Toe)
- trade-off of finding structure vs savings
  - *sometimes* proportional to size of description, not the game tree

Propositional nets

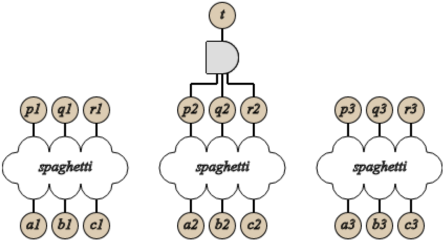
- definition
  - directed bipartite hypergraph
- propositions
- connectives
  - inverter
  - and-gate
  - or-gate
  - transition
- latches, inhibitors, dead state removal

Example propnet

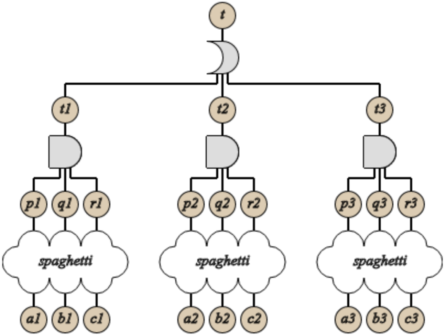


```
role(white)                q :- ~p
base(s)                    r :- true(q)
input(white,a)             r :- does(white,b)
input(white,b)             next(s) :- r
legal(white,a)             goal(white,100) :- true(s)
legal(white,b)             goal(white,0) :- ~true(s)
p :- does(white,a) & true(s)

                           terminal :- true(q)
```

Multiple Buttons and Lights



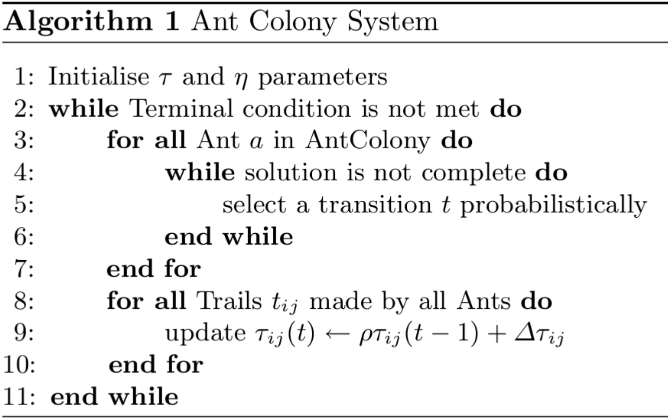
Best Buttons and Lights



General Game Playing with Ants

Ant Colony Optimization

- Marco Dorigo, 1992



Algorithm 1 Ant Colony System

```
Initialise $\tau$ and $\eta$ parameters
while Terminal condition is not met do
    for all Ant $a$ in AntColony do
        while solution is not complete do
            select a transition $t$ probabilistically
        end while
    end for
    for all Trails $t_{ij}$ made by all Ants do
        update $\tau_{ij}(t) \leftarrow \rho\tau_{ij}(t-1) + \Delta\tau_{ij}$
    end for
end while
```

Ant decision propability

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

where

- τ_{xy} is the amount of pheromone deposited for transition from state x to y
- $0 \leq \alpha$ is a parameter to control the influence of τ_{xy}
- η_{xy} is the desirability of state transition xy ("a priori" knowledge)
- $\beta \geq 1$ is a parameter to control the influence of η_{xy}

Pheromon update

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^k$$

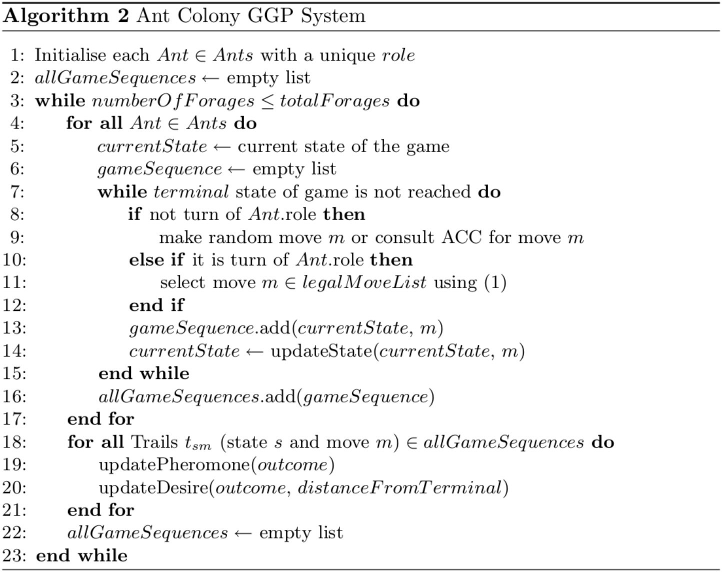
where

τ_{xy} is the amount of pheromone deposited for a state transition *xy*

ρ is the *pheromone evaporation coefficient*

$\Delta\tau_{xy}^k$ is the amount of pheromone deposited by *k*th ant

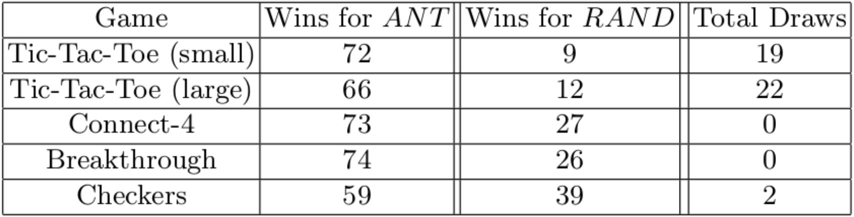
Ants for GGP



Algorithm 2 Ant Colony GGP System

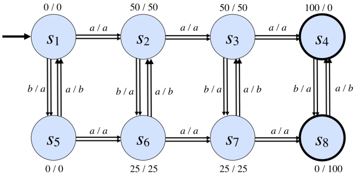
```
Initialise each $Ant \in Ants$ with a unique $role$
$allGameSequences \leftarrow$ empty list
while $numberOfForages \leq totalForages$ do
    for all $Ant \in Ants$ do
        $currentState \leftarrow$ current state of the game
        $gameSequence \leftarrow$ empty list
        while $terminal$ state of game is not reached do
            if not turn of $Ant$.role then
                make random move $m$ or consult ACC for move $m$
            else if it is turn of $Ant$.role then
                select move $m \in legalMoveList$ using (1)
            end if
            $gameSequence$.add($currentState$, $m$)
            $currentState \leftarrow$ updateState($currentState$, $m$)
        end while
        $allGameSequences$.add($gameSequence$)
    end for
    for all Trails $t_{sm}$ (state $s$ and move $m$) $\in allGameSequences$ do
        updatePheromone($outcome$)
        updateDesire($outcome$, $distanceFromTerminal$)
    end for
    $allGameSequences \leftarrow$ empty list
end while
```

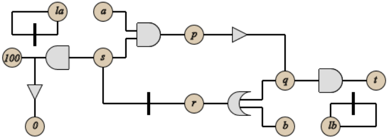
Results

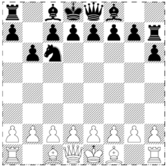


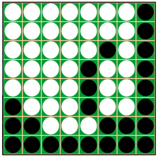
Game	Wins for ANT	Wins for $RAND$	Total Draws
Tic-Tac-Toe (small)	72	9	19
Tic-Tac-Toe (large)	66	12	22
Connect-4	73	27	0
Breakthrough	74	26	0
Checkers	59	39	2

Questions?


0 / 0
50 / 50
50 / 50
100 / 0
s_1
s_2
s_3
s_4
a / a
b / a
a / b
s_5
s_6
s_7
s_8
0 / 0
25 / 25
25 / 25
0 / 100


la
a
p
100
s
q
t
0
r
b
lb





References

- Stanford **General Game Playing** course by Michael Genesereth (`http://www.coursera.org/course/ggp`)
- Michael Genesereth, Michael Thielscher: **General Game Playing** (`http://arrogant.stanford.edu/ggp/chapters/cover.html`)
- Sharma, Shiven, Ziad Kobti, and Scott Goodwin. **"General game playing with ants."** Simulated Evolution and Learning. Springer Berlin Heidelberg, 2008. 381-390.

Image atributions (in order of aperance)

puzzle

> ***Randall Munroe***, `http://xkcd.com/1287/`

GGP competition 1, 2

> `http://cadia.ru.is/wiki/public:cadiaplayer:main#photos`

game manager, state graph, monte carlo, hodgepodge, joint tic tac toe, propnet 1, 2, 3

> ***Michael Genesereth***, `http://arrogant.stanford.edu/ggp/chapters/cover.html`

ants

> ***Mehmet Karatay***, `http://en.wikipedia.org/wiki/File:Safari_ants.jpg`