Decomposition of Multi-Player Games

Dengji Zhao[1] Stephan Schiffel[2] Michael Thielscher[2]

[1]Intelligent Systems Laboratory
University of Western Sydney, Australia

[2]Department of Computer Science
Dresden University of Technology, Germany

AI'09

Outline

Outline

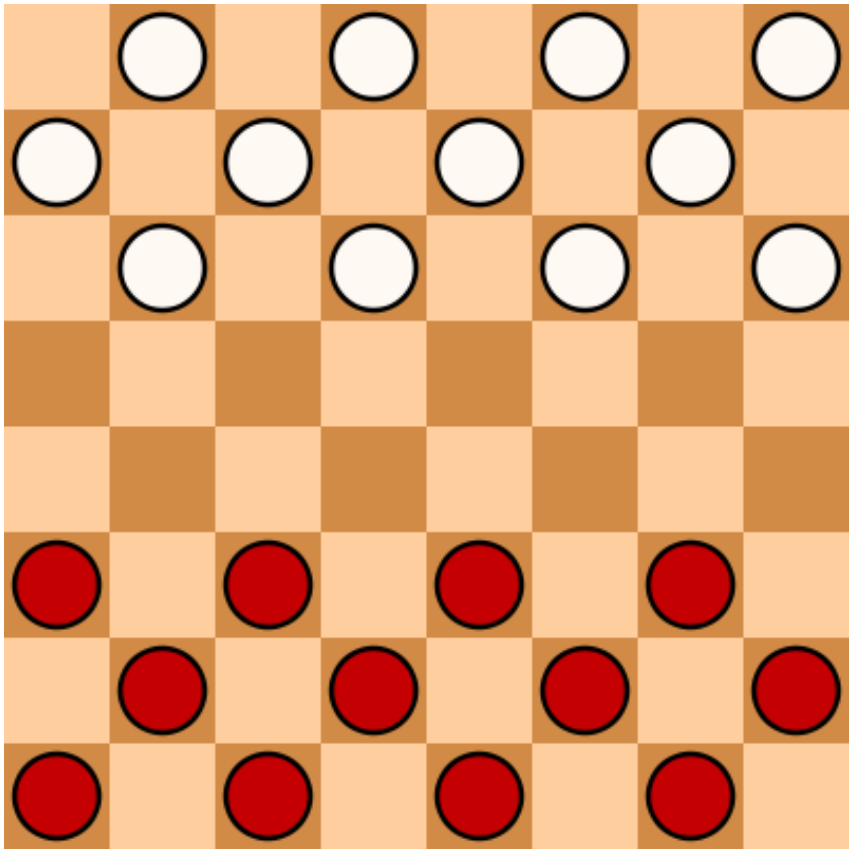
Chess

Chess

Deep Blue beats World Champion (1997)

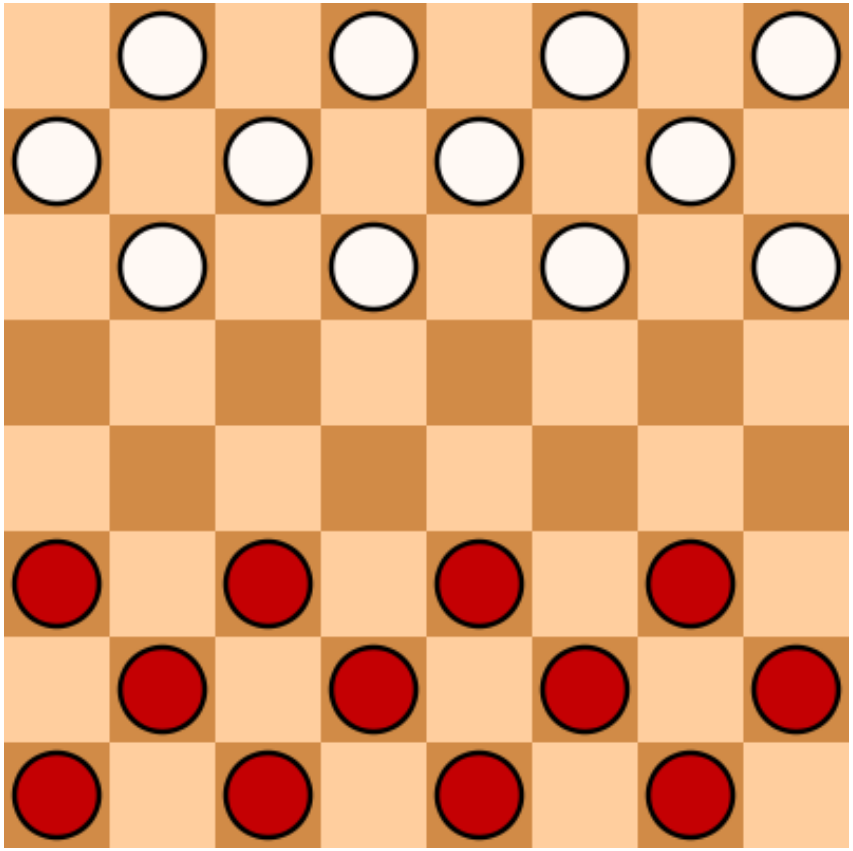
AP / Adam Nadel, File

Checker



Checker

Can Deep Blue play checkers?



General Game Playing

A General Game Player is a system that

- able to accept a formal description of arbitrary games
- able to use such descriptions to play the games effectively

General Game Playing

A General Game Player is a system that

- able to accept a formal description of arbitrary games
- able to use such descriptions to play the games effectively

Constraints:

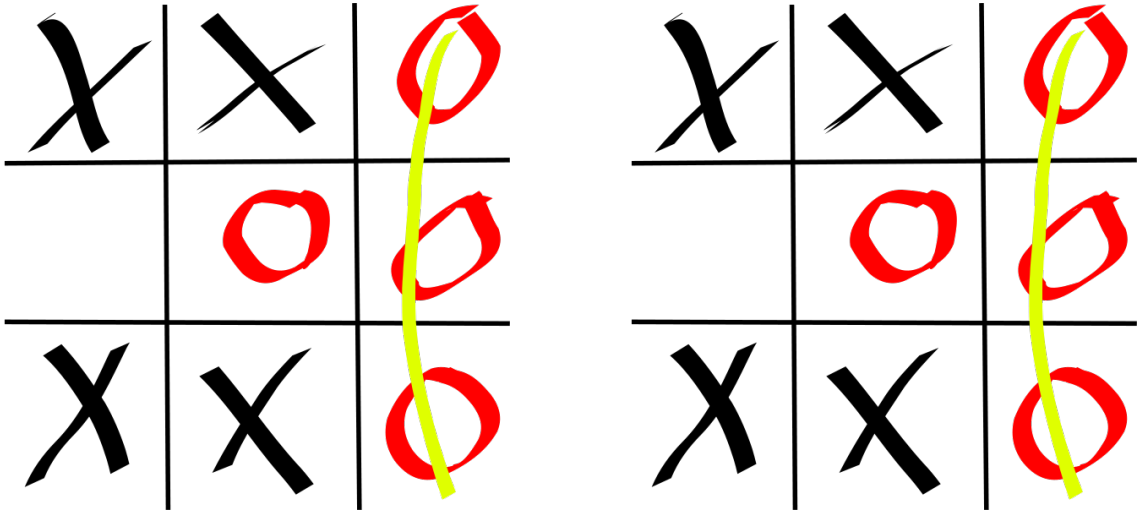
- Time constraints vs. Very large games

Observation:

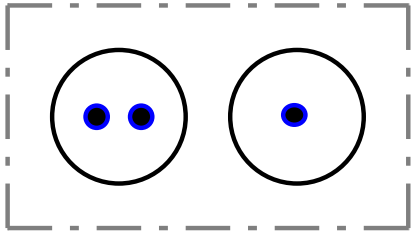
- Games contain independent parts (*subgames*)

Games Contain Subgames

Double-TicTacToe:



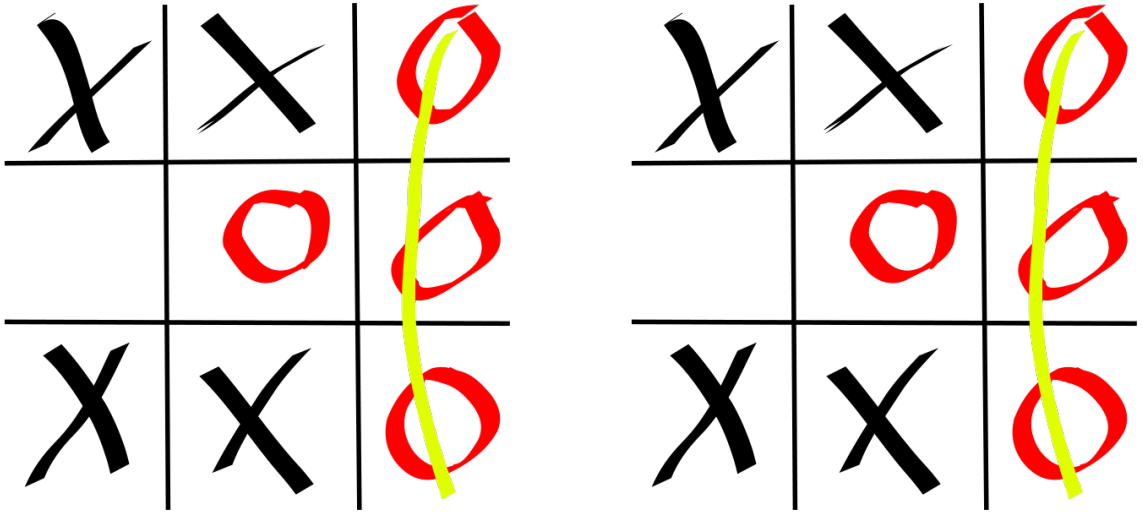
Nim:



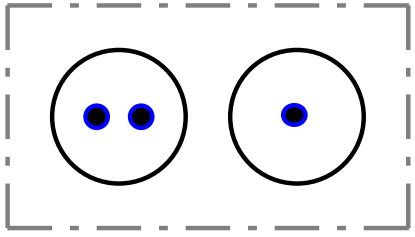
Games Contain Subgames

Improve game search by using their subgames?

Double-TicTacToe:



Nim:



Decomposition

- Widely recognized in AI Planning
- Adapted to General Game Playing
  1. How to decompose?
  2. How to improve search with decomposition?
- Previous Work
  - Single Player Games:
    M. Günther, S. Schiffel, and M. Thielscher: *Factoring general games*. GIGA, 2009

Decomposition

- Widely recognized in AI Planning
- Adapted to General Game Playing
  1. How to decompose?
  2. How to improve search with decomposition?
- Previous Work
  - Single Player Games:
    M. Günther, S. Schiffel, and M. Thielscher: *Factoring general games*. GIGA, 2009

Outline

What is GDL

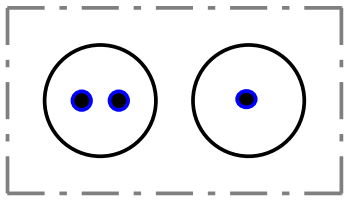
- Variant of Datalog (Prolog)
- Purely axiomatic (*no algebra and arithmetics*)
- Expressive power
  1. n-player ($n \geq 1$)
  2. deterministic
  3. perfect information

How to describe games in DGL

- Player: *role*(xplayer).
- State: set of terms (*fluents*), {cell(1,1,b),...}
  - initial state: *init*(cell(1,1,b)).
- Action: *legal*(Player,Action) $\Leftarrow$ *true*(cell(1,1,b)),...
- Transition: *next*(F) $\Leftarrow$ *does*(...),*true*(...),...
- Termination: *terminal* $\Leftarrow$ line(x).
- Goal/Payoff: *goal*(xplayer, 100) $\Leftarrow$ line(x).

Keywords: *role, init, legal, does, next, terminal, goal, and true.*

Game Nim in GDL



```
(role player1)
(role player2)

(init (heap a 1))
(init (heap b 2))
(init (control player1))

(<= (legal ?p (reduce ?x ?n))
    (true (control ?p))
    (true (heap ?x ?m))
    (smaller ?n ?m))

(<= (next (heap ?x ?n))
    (does ?p (reduce ?x ?n)))
...
(<= terminal
    (true (heap a 0))
    (true (heap b 0)))
(<= (goal ?p 0)
    (true (control ?p)))
...
```

Outline

Definitions

Definition

(Game). A **game** is a tuple $G = (F, A, I, R)$ where

- F is a set of fluents,
- A is a set of actions,
- I is the initial state, a set of ground instances of F,
- R is a set of roles.

Definitions

Definition

(Subgame). A game $G = (F, A, I, R)$ is a **subgame** of $G' = (F', A', I', R')$ iff $F \subseteq F'$, $A \subseteq A'$, $I \subseteq I'$, $R \subseteq R'$, and F, A, I and R are not empty.

Definition

(Subgame Independence). Two subgames $Gs = (Fs, As, Is, Rs)$ and $Gs' = (Fs', As', Is', Rs')$ of game G are **independent** each other iff $Fs \cap Fs' = \oslash$ and $As \cap As' = \oslash$.

Dependency Relations between Fluents and Actions

Potential Precondition (e.g.)

```
(<= (legal ?p (reduce ?x ?n))
    (true (heap ?x ?m))...)
```

(heap ?x ?m) is a precondition of *(reduce ?x ?n)*

Potential Positive Effect (e.g.)

```
(<= (next (heap ?x ?n))
    (does ?p (reduce ?x ?n)))
```

(reduce ?x ?n) is a positive effect of *(heap ?x ?m)*

Potential Negative Effect

F is a negative effect of *M* if *F* is true now, and *F* might be false after execution of *M*

Dependency Relations between Fluents and Actions

Potential Precondition (e.g.)

```
(<= (legal ?p (reduce ?x ?n))
    (true (heap ?x ?m))...)
```

(heap ?x ?m) is a precondition of *(reduce ?x ?n)*

Potential Positive Effect (e.g.)

```
(<= (next (heap ?x ?n))
    (does ?p (reduce ?x ?n)))
```

(reduce ?x ?n) is a positive effect of *(heap ?x ?m)*

Potential Negative Effect

F is a negative effect of *M* if *F* is true now, and *F* might be false after execution of *M*

Dependency Relations between Fluents and Actions

Potential Precondition (e.g.)

```
(<= (legal ?p (reduce ?x ?n))
    (true (heap ?x ?m))...)
```

(heap ?x ?m) is a precondition of *(reduce ?x ?n)*

Potential Positive Effect (e.g.)

```
(<= (next (heap ?x ?n))
    (does ?p (reduce ?x ?n)))
```

(reduce ?x ?n) is a positive effect of *(heap ?x ?m)*

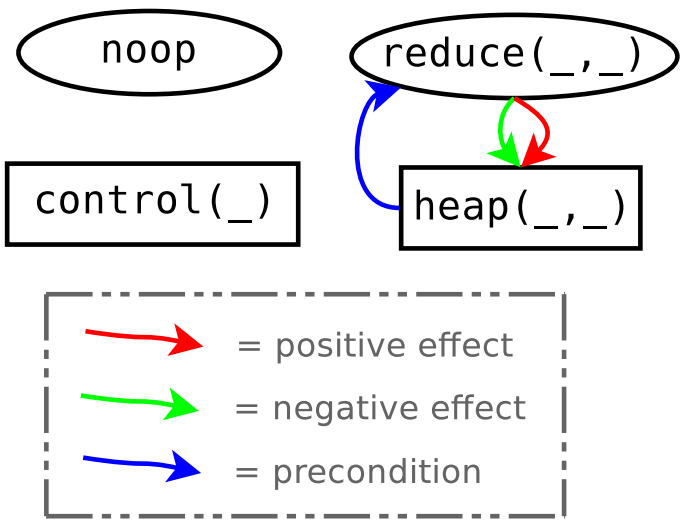
Potential Negative Effect

F is a negative effect of *M* if *F* is true now, and *F* might be false after execution of *M*

Finding Independent Subgames

Subgames

connected components of fluents and actions

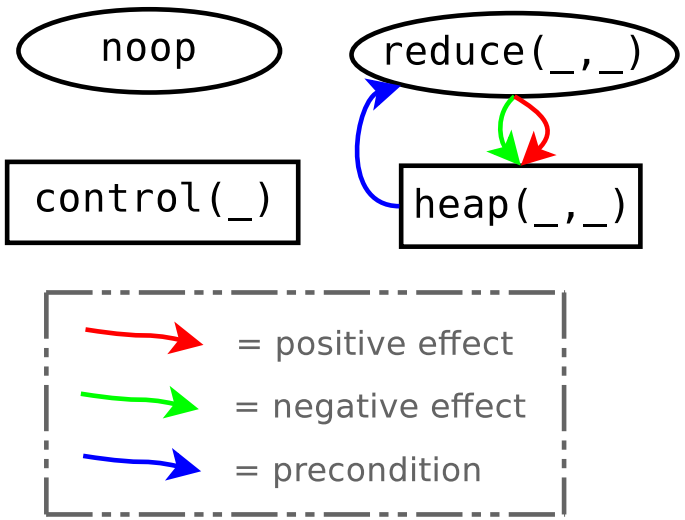


Finding Independent Subgames

Problem

subgames share fluent and action names

- obvious subgames (*heap a* and *heap b*) are still connected

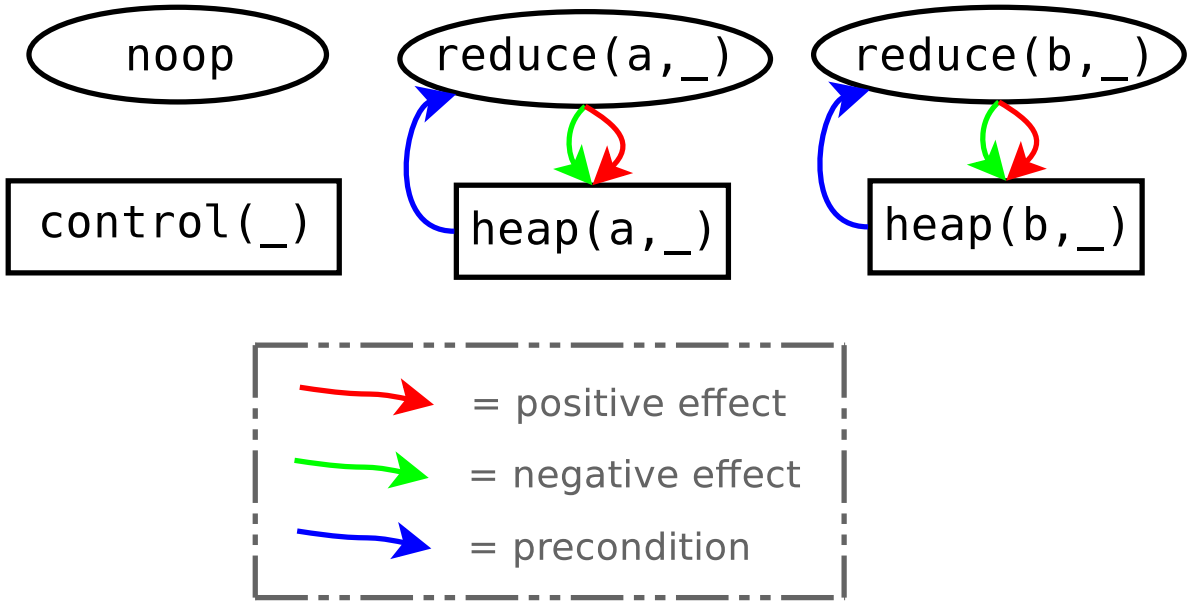


Finding Independent Subgames

Solution

argument instantiation

- *fluent*: argument's value does NOT change in the whole game
- *move*: refers to an instantiated argument of a fluent



Outline

Decomposition Search

Main Idea

1. search subgames separately *(Subgame Search)*
2. build global plans with subgame plans *(Global Game Search)*

Problem

- *goal* and *terminal* conditions are NOT defined for subgames

Decomposition Search

Main Idea

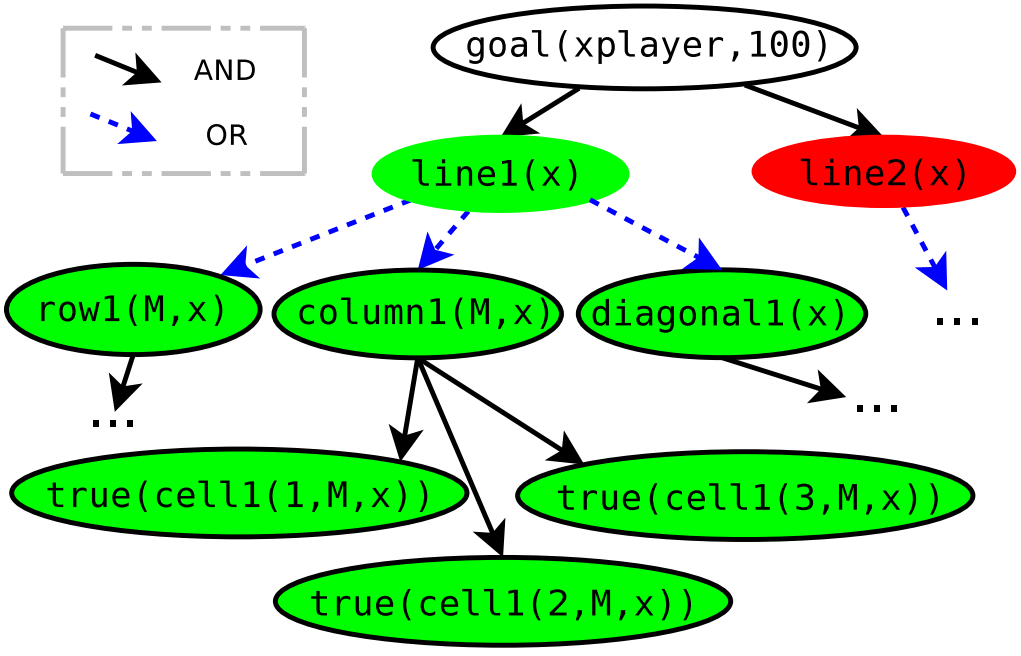
1. search subgames separately *(Subgame Search)*
2. build global plans with subgame plans *(Global Game Search)*

Problem

- *goal* and *terminal* conditions are NOT defined for subgames

Local Concept *(goal and terminal rule decomposition)*

Local Concept detection example (*double-tictactoe*):
(<= (goal xplayer 100) (line1 x) (line2 x))

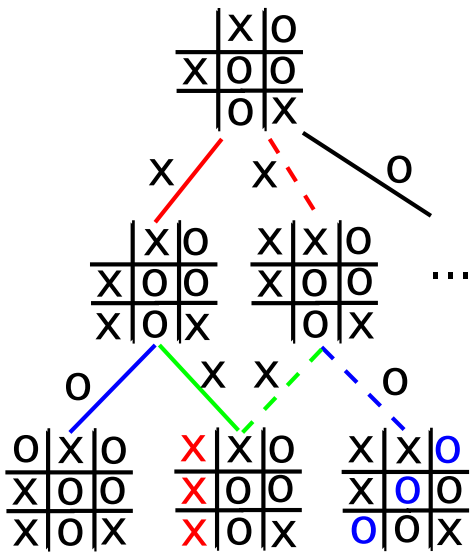


Local Concepts: line1(x), line2(x)

Turn-Move Sequence

Turn-Move Sequence

a sequence of pairs *(Player,Action)* (*a path in subgame tree*) + a set of evaluations of local concepts in the subgame



4 turn-move sequences for *xplayer*:

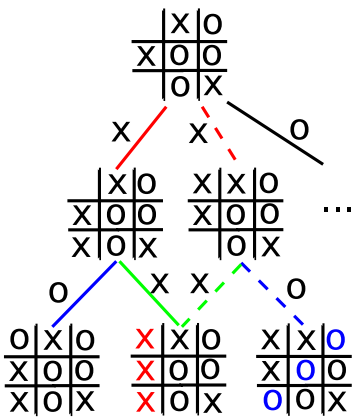
$[(x, mark(3,1)) \circ (o, mark(1,1)), Eval1]$
$[(x, mark(3,1)) \circ (x, mark(1,1)), Eval2]$
$[(x, mark(1,1)) \circ (x, mark(3,1)), Eval3]$
$[(x, mark(1,1)) \circ (o, mark(3,1)), Eval4]$

The Target of Subgame Search

Sequence Simplification

remove evaluation dominated sequences (*paths*)

- a seq is evaluation dominated if there is another similar seq with better evaluations
- a set of seqs S start with a move of player p is removed
  - if all $s \in S$ are dominated by other seqs start with other moves of p



$[(x, mark(3,1)) \circ (o, mark(1,1)), Eval1]$
$[(x, mark(3,1)) \circ (x, mark(1,1)), Eval2]$
The following two are dominated by the above:
$[(x, mark(1,1)) \circ (x, mark(3,1)), Eval3]$
$[(x, mark(1,1)) \circ (o, mark(3,1)), Eval4]$

Simple Example

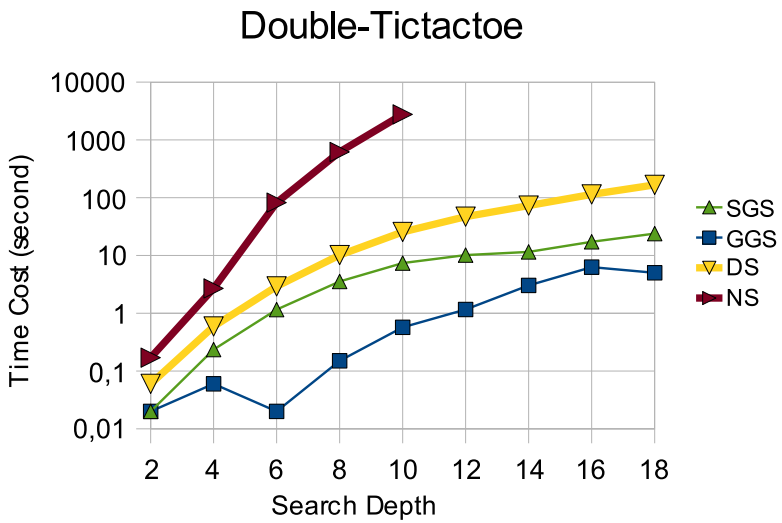
Using normal search methods, for each state,

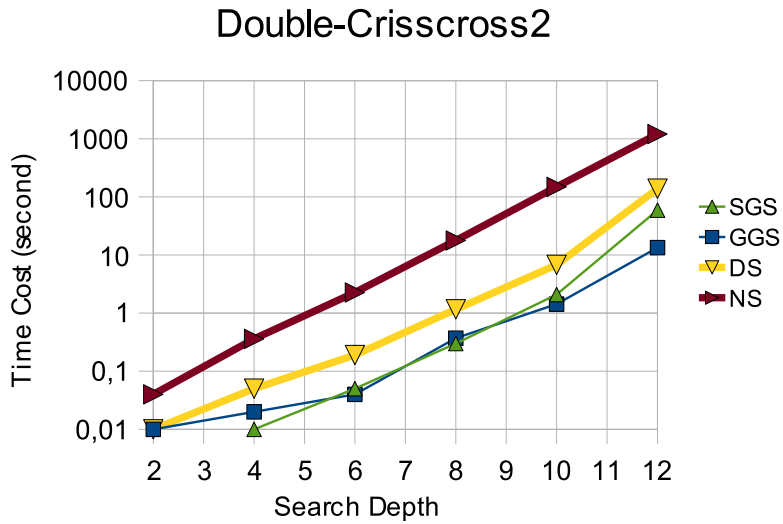
- choose legal moves from turn-move sequences returned from subgame search

Outline

Testing Results Comparison

time cost(second): for finding the first optimal strategy
DS: Decomposition Search; *NS*: Normal Search;
SGS: Subgame Search; *GGS*: Global Game Search





Done and ToDo

What we have done:

1. subgame detection
2. decomposition search (incl. a special version for impartial games)

Done and ToDo

What we have done:

1. subgame detection
2. decomposition search (incl. a special version for impartial games)

What can be improved:

1. more efficient subgame detection method
2. apply pruning techniques in decomposition search
3. use subgame plans more efficiently in global game search

Q&A

Thank you for your attention!

For Further Reading I


John H. Conway
On Numbers and Games.
Academic Press, 1976.


Elwyn R. Berlekamp, John H. Conway, Richard K. Guy
Winning Ways 2nd Edition.
2001.


Martin Müller
Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames
1999.

For Further Reading II


M. Günther, S. Schiffel and M. Thielscher
Decomposition of Single Player Games
2007.


Eric Schkufza
Decomposition of Games for Efficient Reasoning
2008.

Time Complexity Comparison I

Impartial and Partial Games

Assume that a game G has n subgames, G_1, G_2, ..., G_n with V_1, V_2, ..., V_n states respectively,

- normal search: $O(V_1 * V_2 * ... * V_n)$
- decomposition search: $O(V_1 + V_2 + ... + V_n)$

Example

For double-tictactoe, the number of states is about 18!(including revisited states), while the state for each subgame is about $\prod_{n=1}^{9}(2n)$ which is $\prod_{n=1}^{9}(2n-1)$ times smaller than 18!

Time Complexity Comparison II

Parallel Games

Assume that a parallel game G has n subgames, G_1, G_2, ..., G_n with V_1, V_2, ..., V_n states respectively,

- normal search: $O(V_1 * V_2 * ... * V_n)$
- decomposition search: $O(V_1 + V_2 + ... + V_n)$

Serial Games

Assume that a serial game G has n subgames, for subgame i there are V_i states and T_i terminal states

- normal search:
  $O(V_1 + T_1 * V_2 + T_1 * T_2 * V_3 + ... + T_1 * T_2 * ... * T_{n-1} * V_n)$
- decomposition search: $O(V_1 + V_2 + ... + V_n)$