

Content

- 1 Gomoku Rule
- 2 Evolutionary Algorithm
- 3 Reinforcement Learning
- 4 Monte Carlo Tree Search
- 5 Proof-number Search
- 6 Dependency-based Search
- 7 Solve Gomoku
- 8 Alpha-beta Pruning
- 9 Gomoku Competition

Gomoku Rule

- Gomoku:
 - Free Gomoku Rule: *Victoria* $15 \times 15[1][2][3]$
 - Standard Gomoku Rule: *Victoria* 15×15

Gomoku Rule

- Gomoku:
 - Free Gomoku Rule: *Victoria* $15 \times 15[1][2][3]$
 - Standard Gomoku Rule: *Victoria* 15×15
 - Swap 2 Rule

Gomoku Rule

- Gomoku:
 - Free Gomoku Rule: *Victoria* 15×15 [1][2][3]
 - Standard Gomoku Rule: *Victoria* 15×15
 - Swap 2 Rule
- Renju:
 - Free Renju Rule[4]

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Gomoku:
 - Free Gomoku Rule: *Victoria* 15×15 [1][2][3]
 - Standard Gomoku Rule: *Victoria* 15×15
 - Swap 2 Rule
- Renju:
 - Free Renju Rule[4]
 - Soosyrv-8 Rule

Evolutionary Algorithm

Evolutionary Algorithm

- Genetic Algorithm:solutions are in the form of strings[5]

Evolutionary Algorithm

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

- Genetic Algorithm: solutions are in the form of strings[5]
- Genetic Programming: solutions are in the form of computer programs[6][7]

Evolutionary Algorithm

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

- Genetic Algorithm: solutions are in the form of strings[5]
- Genetic Programming: solutions are in the form of computer programs[6][7]
- Evolutionary Programming: only numerical parameters of programs are allowed to evolve[8]

Evolutionary Algorithm

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

- Genetic Algorithm: solutions are in the form of strings[5]
- Genetic Programming: solutions are in the form of computer programs[6][7]
- Evolutionary Programming: only numerical parameters of programs are allowed to evolve[8]
- Neuroevolution: genomes represent artificial neural networks[9][10]
- ...

Evolutionary Algorithm

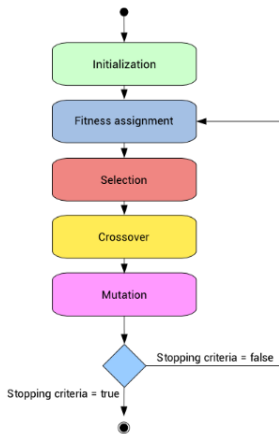


Figure: Flow chart of evolutionary algorithm

Jitong Qi

Gomoku Rule

**Evolutionary
Algorithm**

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

Genetic Algorithm[5]

Genetic Algorithm[5]

- Coding Scheme: coordinates of consecutive seven steps
Representation: $[(9,4), (7,6), (6,4), (7,7), (8,4), (7,4), (7,8)]$

Genetic Algorithm[5]

- Coding Scheme: coordinates of consecutive seven steps
Representation: $[(9,4), (7,6), (6,4), (7,7), (8,4), (7,4), (7,8)]$
- Fitness function

Genetic Algorithm[5]

- Coding Scheme: coordinates of consecutive seven steps
Representation: $[(9,4), (7,6), (6,4), (7,7), (8,4), (7,4), (7,8)]$
- Fitness function
- Selection: Roulette Wheel Selection

- Crossover: random crossover point

Parent 1: (7,6) (5,5) (5,4) (2,2) (6,5) (7,7) (4,4)

Parent 2: (5,7) (5,5) (2,4) (7,7) (3,5) (4,4) (3,3)

Child 1: (7,6) (5,5) (5,4) (7,7) (3,5) (4,4) (3,3)

Child 2: (5,7) (5,5) (2,4) (2,2) (6,5) (7,7) (4,4)

- Crossover: random crossover point

Parent 1: (7,6) (5,5) (5,4) (2,2) (6,5) (7,7) (4,4)

Parent 2: (5,7) (5,5) (2,4) (7,7) (3,5) (4,4) (3,3)

Child 1: (7,6) (5,5) (5,4) (7,7) (3,5) (4,4) (3,3)

Child 2: (5,7) (5,5) (2,4) (2,2) (6,5) (7,7) (4,4)

- Mutate: random mutate point

Parent 3: (7,6) (5,5) (5,4) (2,2) (6,5) (7,7) (4,4)

Child 3: (7,6) (5,5) (5,4) (1,1) (6,5) (7,7) (4,4)

Jitong Qi

Gomoku Rule

Evolutionary
Algorithm

**Reinforcement
Learning**

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

Reinforcement Learning

Reinforcement Learning

- Temporal difference learning:[11][12]

Reinforcement Learning

- Temporal difference learning:[11][12]
- Q-learning

Reinforcement Learning

- Temporal difference learning:[11][12]
- Q-learning
- Deep Q-learning

Monte Carlo Tree Search[13][14][15]

- Selection
- Play-out
- Expansion
- Backpropagation

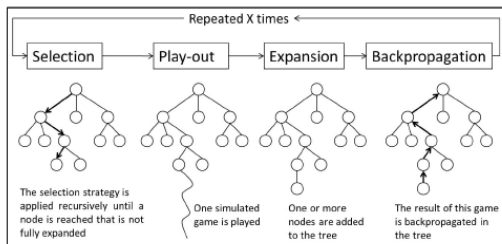


Figure: Flow chart of MCTS

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

**Monte Carlo
Tree Search**

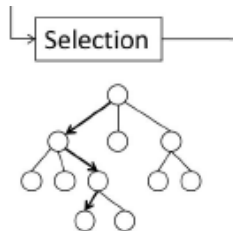
Proof-number
Search

Dependency-
based
Search

Solve Gomoku

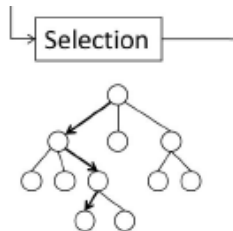
Alpha-beta
Pruning

Gomoku
Competition



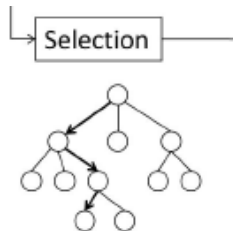
The selection strategy is applied recursively until a node is reached that is not fully expanded

- Traverse down the tree from the root, according to a selection policy



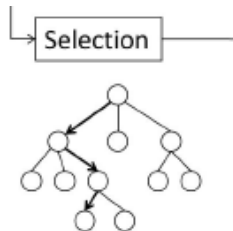
The selection strategy is applied recursively until a node is reached that is not fully expanded

- Traverse down the tree from the root, according to a selection policy
- Until a node is reached that is not fully expanded

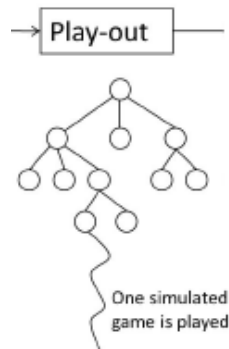


The selection strategy is applied recursively until a node is reached that is not fully expanded

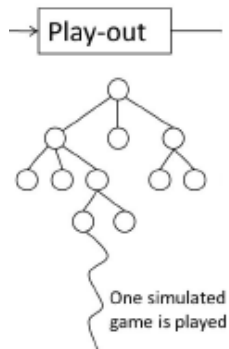
- Traverse down the tree from the root, according to a selection policy
- Until a node is reached that is not fully expanded
- Trade-off between exploration and exploitation



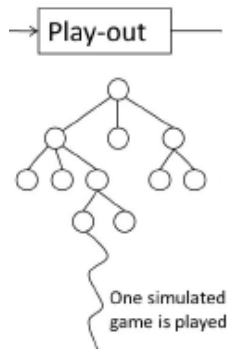
The selection strategy is applied recursively until a node is reached that is not fully expanded



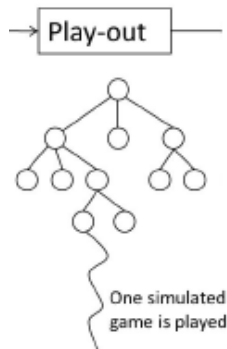
- Random game is played



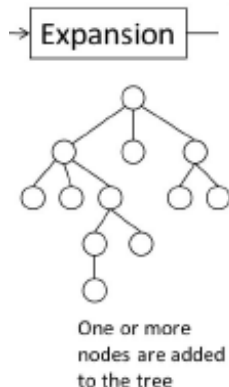
- Random game is played
- Roll-out policy



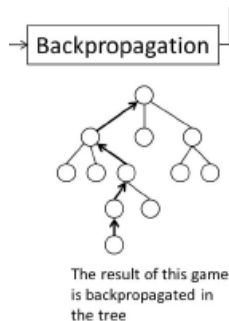
- Random game is played
- Roll-out policy
- Limited depth



- Expand the game tree with all nodes in play-out step



- Back propagate outcome and visit numbers



Jitong Qi

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

**Proof-number
Search**

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

Solve Game

Solve Game

- Choose a knowledge re-presentation

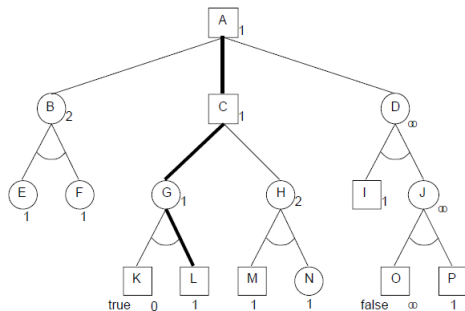
Solve Game

- Choose a knowledge re-presentation
- Perform a search: single-agent tree, game trees, AND/OR tree

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

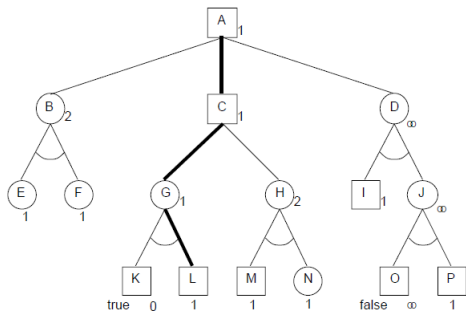
Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

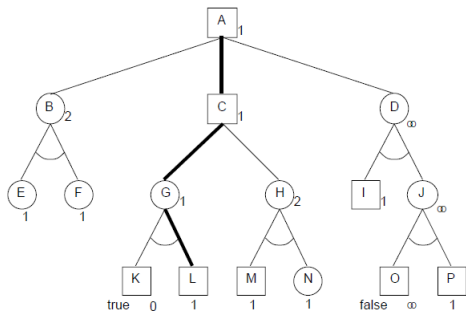
Alpha-beta
PruningGomoku
Competition

- Node value: TRUE, FALSE, UNKNOWN

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

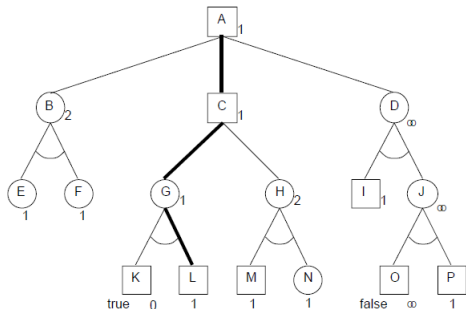
Alpha-beta
PruningGomoku
Competition

- Node value: TRUE,FALSE,UNKNOWN
- circle:AND, square:OR

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

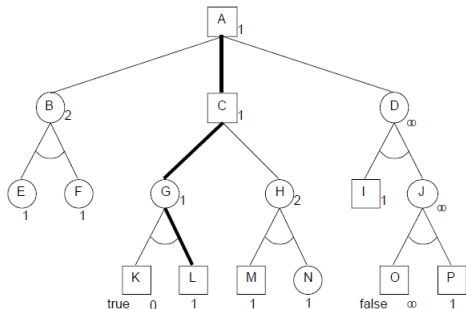
Alpha-beta
PruningGomoku
Competition

- circle:AND, square:OR

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

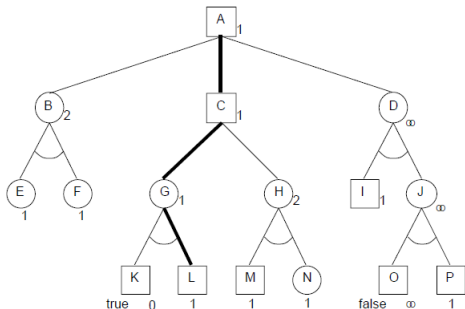
Alpha-beta
PruningGomoku
Competition

- For any AND/OR tree T a set of frontier nodes S is a proof set if proving all nodes within S proves T
- circle:AND, square:OR

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

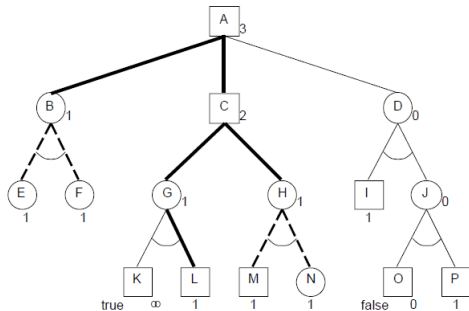
- For any AND/OR tree T a set of frontier nodes S is a proof set if proving all nodes within S proves T
- For any AND/OR tree T , the proof number of T is defined as the cardinality of the smallest proof set of T .
- circle:AND, square:OR

Disproof Number

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

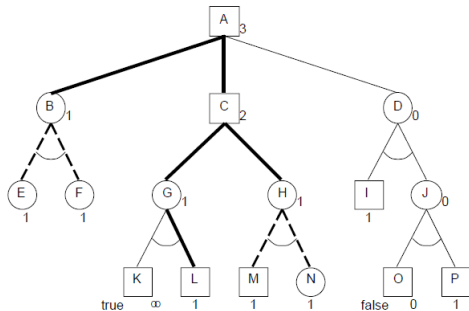
- circle:AND, square:OR

Disproof Number

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

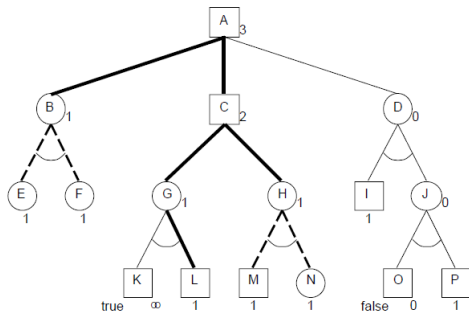
- For any AND/OR tree T a set of frontier nodes S is a disproof set if disproving all nodes within S disproves T
- circle:AND, square:OR

Disproof Number

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

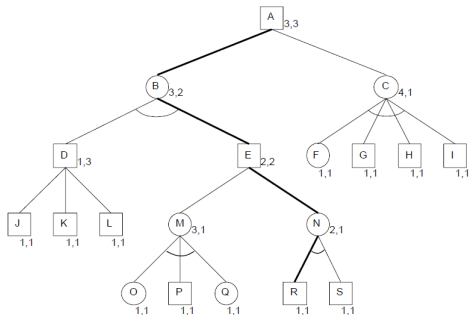
- For any AND/OR tree T a set of frontier nodes S is a disproof set if disproving all nodes within S disproves T
- For any AND/OR tree T , the disproof number of T is defined as the cardinality of the smallest disproof set of T .
- circle:AND, square:OR

Proof Number Search[3][16]

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree Search
**Proof-number
Search**
Dependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

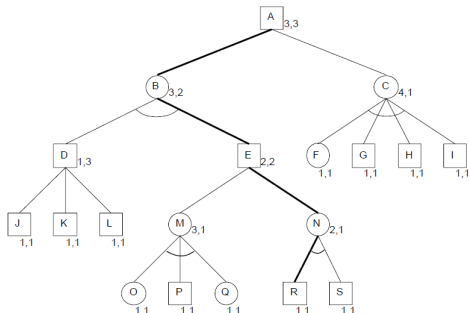
- circle:AND, square:OR

Proof Number Search[3][16]

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

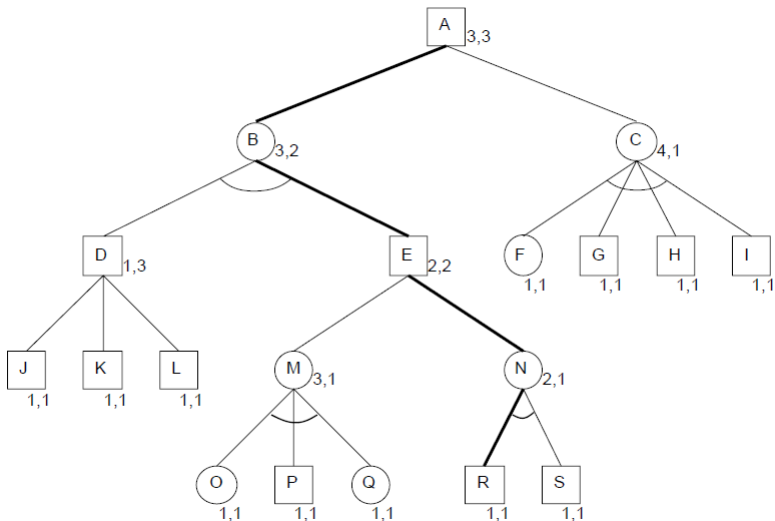
- For any AND/OR tree T , a most-proving node of T is a frontier node of T , which by obtaining the value `TRUE` reduces T 's proof number by 1, while by obtaining the value `FALSE` reduces T 's disproof number by 1.
- circle:AND, square:OR

Proof Number Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Dependency-based Search

Example: a production system consisting 10 rewriting rules

r_0, \dots, r_9

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Example: a production system consisting 10 rewriting rules

r_0, \dots, r_9

- $0 \xrightarrow{r_0} A, 1 \xrightarrow{r_0} B, 2 \xrightarrow{r_0} C, 3 \xrightarrow{r_0} D, 4 \xrightarrow{r_0} E, 5 \xrightarrow{r_0} F,$
 $6 \xrightarrow{r_0} G, 7 \xrightarrow{r_0} H, 8 \xrightarrow{r_8} I, 9 \xrightarrow{r_9} J$

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Example: a production system consisting 10 rewriting rules

r_0, \dots, r_9

- $0 \xrightarrow{r_0} A, 1 \xrightarrow{r_0} B, 2 \xrightarrow{r_0} C, 3 \xrightarrow{r_0} D, 4 \xrightarrow{r_0} E, 5 \xrightarrow{r_0} F,$
 $6 \xrightarrow{r_0} G, 7 \xrightarrow{r_0} H, 8 \xrightarrow{r_8} I, 9 \xrightarrow{r_9} J$
- $ABCDEFGHIJ \xrightarrow{r_{10}} X$

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Example: a production system consisting 10 rewriting rules

r_0, \dots, r_9

- $0 \xrightarrow{r_0} A, 1 \xrightarrow{r_0} B, 2 \xrightarrow{r_0} C, 3 \xrightarrow{r_0} D, 4 \xrightarrow{r_0} E, 5 \xrightarrow{r_0} F,$
 $6 \xrightarrow{r_0} G, 7 \xrightarrow{r_0} H, 8 \xrightarrow{r_8} I, 9 \xrightarrow{r_9} J$
- $ABCDEFGHIJ \xrightarrow{r_{10}} X$
- Initial state: 0123456789

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Example: a production system consisting 10 rewriting rules

r_0, \dots, r_9

- $0 \xrightarrow{r_0} A, 1 \xrightarrow{r_0} B, 2 \xrightarrow{r_0} C, 3 \xrightarrow{r_0} D, 4 \xrightarrow{r_0} E, 5 \xrightarrow{r_0} F,$
 $6 \xrightarrow{r_0} G, 7 \xrightarrow{r_0} H, 8 \xrightarrow{r_8} I, 9 \xrightarrow{r_9} J$
- $ABCDEFGHIJ \xrightarrow{r_{10}} X$
- Initial state: 0123456789
- Goal: X

Jitong Qi

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

**Dependency-
based
Search**

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

Dependency-based Search

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Attributes:

$$U = \{A(i, j, z) | 1 \leq i \leq j \leq 10, z \in \{A, \dots, J, 0, \dots, 9, X\}\}$$

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Attributes:

$$U = \{A(i, j, z) | 1 \leq i \leq j \leq 10, z \in \{A, \dots, J, 0, \dots, 9, X\}\}$$

- Initial state

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Attributes:

$$U = \{A(i, j, z) | 1 \leq i \leq j \leq 10, z \in \{A, \dots, J, 0, \dots, 9, X\}\}$$

- Initial state
- Goal state: $\{A(1, 10, X)\}$

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Attributes:

$$U = \{A(i, j, z) | 1 \leq i \leq j \leq 10, z \in \{A, \dots, J, 0, \dots, 9, X\}\}$$

- Initial state

- Goal state: $\{A(1, 10, X)\}$

- Operator U_f : f as a 3-tuple

$$\langle f^{pre}, f^{del}, f^{add} \rangle, f^{pre}, f^{del}, f^{add} \subset U \text{ and } f^{del} \subset f^{pre}$$

- $r_0 = \langle \{A(1, 1, 0)\}, \{A(1, 1, 0)\}, \{A(1, 1, A)\} \rangle$

Jitong Qi

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

**Dependency-
based
Search**

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

Dependency-based Search

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Paths U_p : Series of operators applicable to initial state $\{r_0, r_1\}, \{r_0, r_1, r_2\}$ is a path; $\{r_0, r_0\}$ isn't a path

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Paths U_p : Series of operators applicable to initial state $\{r_0, r_1\}, \{r_0, r_1, r_2\}$ is a path; $\{r_0, r_0\}$ isn't a path
- Paths equivalent $P \equiv Q$: P is a permutation of Q and P, Q are paths $\{r_0, r_1\} \equiv \{r_1, r_0\}$

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Paths U_p : Series of operators applicable to initial state $\{r_0, r_1\}, \{r_0, r_1, r_2\}$ is a path; $\{r_0, r_0\}$ isn't a path
- Paths equivalent $P \equiv Q$: P is a permutation of Q and P, Q are paths $\{r_0, r_1\} \equiv \{r_1, r_0\}$
- Path class: $U_p / \equiv = \overline{\{r_0, r_1\}}$

Dependency-based Search

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- Paths U_p : Series of operators applicable to initial state $\{r_0, r_1\}, \{r_0, r_1, r_2\}$ is a path; $\{r_0, r_0\}$ isn't a path
- Paths equivalent $P \equiv Q$: P is a permutation of Q and P, Q are paths $\{r_0, r_1\} \equiv \{r_1, r_0\}$
- Path class: $U_p / \equiv: \overline{\{r_0, r_1\}}$
- Key class U_k : $\overline{\{r_0\}}, \dots, \overline{\{r_9\}}, \overline{\{r_0, \dots, r_9, r_{10}\}}$
- $\overline{\{r_0, r_1\}}$ is a class but not a key class.

Solve Gomoku

Method: pn-search + db-search[17]

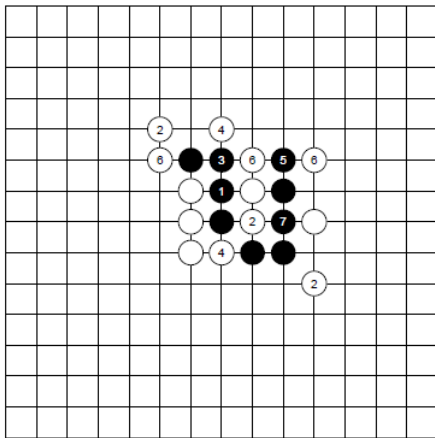
Solve Gomoku

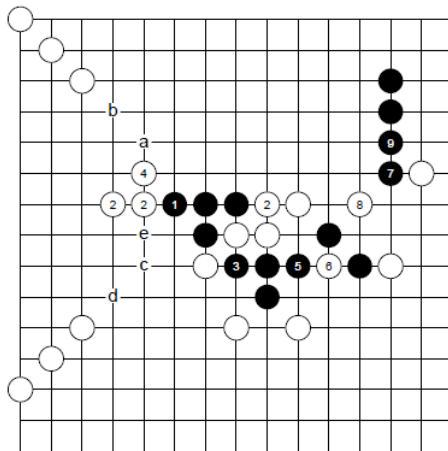
Method: pn-search + db-search[17]

- Attributes
- Initial state
- Goal state

Method: pn-search + db-search[17]

- Attributes
- Initial state
- Goal state
- Operator:
transform into
single-agent





Minimax Algorithm

```

function MINIMAX-DECISION(state) returns an action
  return  $\operatorname{argmax}_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(\text{state}, a))$ 

```

```

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow -\infty$ 
  for each a in ACTIONS(state) do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
  return v

```

```

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow \infty$ 
  for each a in ACTIONS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
  return v

```

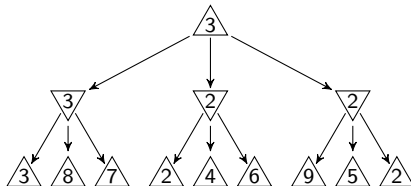
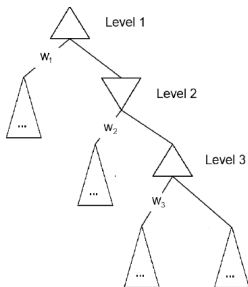


Figure: Pseudocode of minimax algorithm

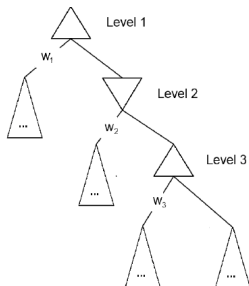
Meaning of α and β

- Consider a modification of alpha-beta pruning where you keep a list containing the best value, w_i , for the minimizer/maximizer (depending on the level) at each level up to and including the current level.
- Assume that the root node is always a max node.



Meaning of α and β

- Consider a modification of alpha-beta pruning where you keep a list containing the best value, w_i , for the minimizer/maximizer (depending on the level) at each level up to and including the current level.
- Assume that the root node is always a max node.
- What is the relationship between α, β and the list of w_1, \dots, w_n at a max node at the n th level of the tree

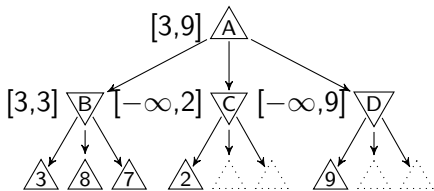


Meaning of α and β

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

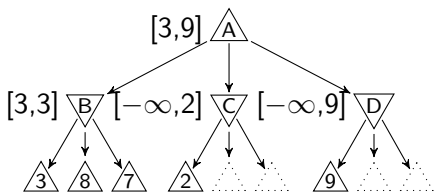
Alpha-beta
PruningGomoku
Competition

Meaning of α and β

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

$[\alpha, \beta]$ is a closed interval
as the estimation of
utility value so far.

Pseudocode of alpha-beta pruning

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
return the *action* in $\text{ACTIONS}(\text{state})$ with value v

function MAX-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow +\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

Figure: Pseudocode of alpha-beta pruning

Pseudocode of alpha-beta pruning

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
return the *action* in $\text{ACTIONS}(\text{state})$ with value v

function MAX-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow +\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

- Each node inherit $[\alpha, \beta]$ from its parent node

Figure: Pseudocode of alpha-beta pruning

Pseudocode of alpha-beta pruning

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
return the *action* in $\text{ACTIONS}(\text{state})$ with value v

function MAX-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow +\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

- Each node inherit $[\alpha, \beta]$ from its parent node
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}

Figure: Pseudocode of alpha-beta pruning

Pseudocode of alpha-beta pruning

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
return the *action* in $\text{ACTIONS}(\text{state})$ with value v

function MAX-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow +\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

- Each node inherit $[\alpha, \beta]$ from its parent node
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

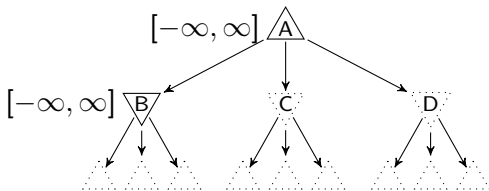
Figure: Pseudocode of alpha-beta pruning

Example of alpha-beta pruning1

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

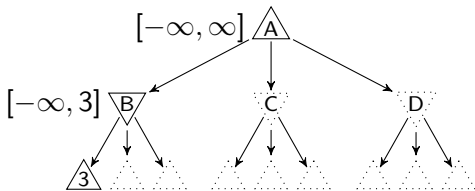
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning2

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

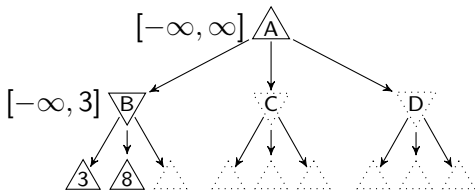
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning3

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

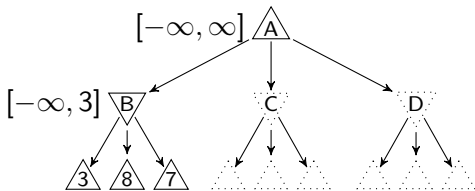
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning4

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

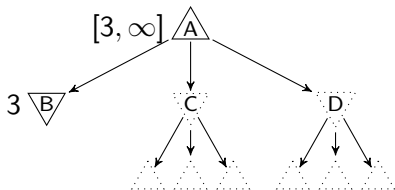
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning5

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

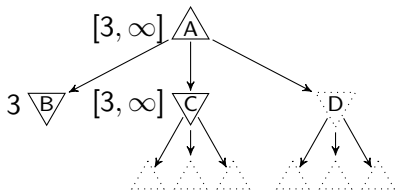
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning6

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

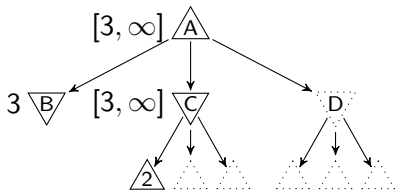
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning7

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

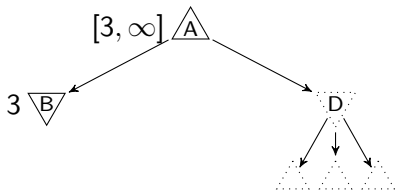
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning8

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

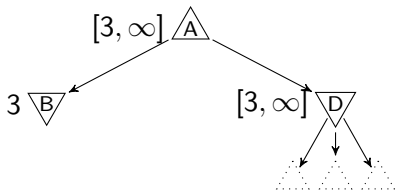
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

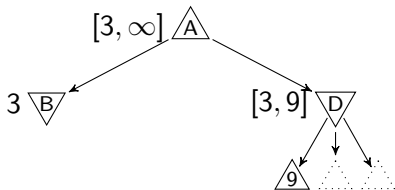
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning10

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

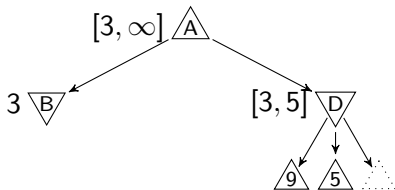
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

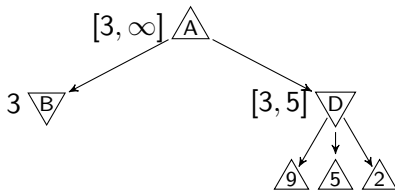
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning¹²

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

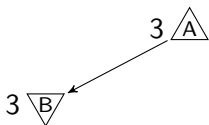
- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Example of alpha-beta pruning13

Gomoku Rule

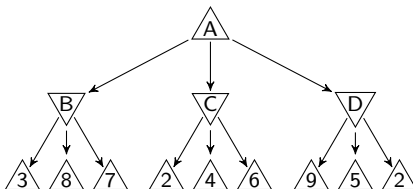
Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

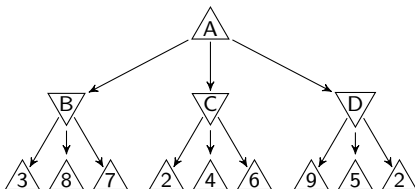
Alpha-beta
PruningGomoku
Competition

- For MAX node, β is fixed as β_{parent} , v is used to update α initialized as α_{parent}
- For MIN node, α is fixed as α_{parent} , v is used to update β initialized as β_{parent}

Input of Practice

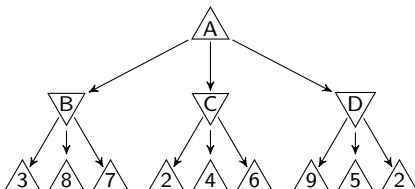


Input of Practice



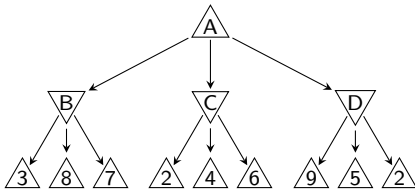
- First line: The role of the root node(1 for MAX node and 0 for MIN node) and the depth of the tree
- Example: 1 3

Input of Practice



- First line: The role of the root node(1 for MAX node and 0 for MIN node) and the depth of the tree
- Example: 1 3
- Second line: Tree Structure
- Example: $[[3,8,7],[2,4,6],[9,5,2]]$

Output of Practice

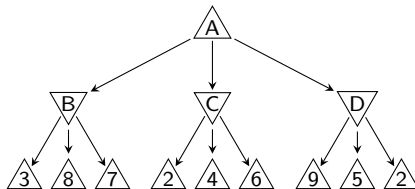


Output of Practice

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

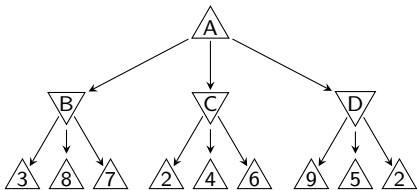
- First line: The result for minimax search
- Example: 3

Output of Practice

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

- First line: The result for minimax search
- Example: 3
- Second line: Pruned nodes in order
- Example: 4 6

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

- Gomoku Rule: Standard Gomoku Rule(exact five)
- Balanced opening from a set
- The board has 20x20 squares
- 30 seconds per move, 180 seconds per match
- 10 fixed AI, 12 matches with each one
- 3 points for a win

Gomoku Competition

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

- No more than 3 people a team
- Upload to test once a day
- Uploading file may not take more than 5 MB of space (zipped together with all necessary files) named as id.zip
- Only one CPU core
- No deep neural network
- No pondering

Gomoku Competition

Gomoku Rule

Evolutionary
Algorithm

Reinforcement
Learning

Monte Carlo
Tree Search

Proof-number
Search

Dependency-
based
Search

Solve Gomoku

Alpha-beta
Pruning

Gomoku
Competition

- `ftp://10.88.3.60`
- Gomoku manager
`http://gomocup.org/download-gomocup-manager/`
- AI `http://gomocup.org/download-gomoku-ai/`
- Python Template
`https://github.com/stranskyjan/pbrain-pyrandom`
- Gomocup `http://gomocup.org/`

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Louis Victor Allis and Hj Van Den Herik.
Go-moku and threat-space search.
. Ist. Psu. Edu/, 1993.



Louis Victor Allis, H. Jaap van den Herik, and M. P. H. Huntjens.
Go-Moku Solved By New Search Techniques.
IEEE Comput. Intell. Mag., 12(1):7–23, 1996.



Louis Victor Allis.
Searching for Solutions in Games and Artificial Intelligence.
1994.



Wagner Jamos and Virag Istvan.
Solving renju.
ICGA J., (March):30–35, 2001.



Junru Wang and Lan Huang.
Evolving Gomoku Solver by Genetic Algorithm.
pages 1064–1067, 2014.



Atif M. Alhejali and Simon M. Lucas.
Evolving diverse Ms. Pac-Man playing agents using genetic programming.
2010 UK Work. Comput. Intell. UKCI 2010, (April), 2010.

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Atif M. Alhejali and Simon M. Lucas.

Using genetic programming to evolve heuristics for a Monte Carlo Tree Search Ms Pac-Man agent.

IEEE Conf. Comput. Intell. Games, CIG, 2013.



Marcus Gallagher and A Ryan.

Learning to play pac-man: An evolutionary rule based-approach.

Proc, 03:2462–2469, 2003.



Keunhyun Oh and Sung Bae Cho.

A hybrid method of Dijkstra algorithm and evolutionary neural network for optimal Ms. Pac-Man agent.

Proc. - 2010 2nd World Congr. Nat. Biol. Inspired Comput. NaBIC 2010, pages 239–243, 2010.



Simon M. Lucas.

Evolving a neural network location evaluator to play ms. pac-man.

IEEE Symp. Comput. Intell. ..., pages 203–210, 2005.



Dongbin Zhao, Zhen Zhang, and Yujie Dai.

Self-teaching adaptive dynamic programming for Gomoku.

Neurocomputing, 78(1):23–29, 2012.

Gomoku Rule

Evolutionary
AlgorithmReinforcement
LearningMonte Carlo
Tree SearchProof-number
SearchDependency-
based
Search

Solve Gomoku

Alpha-beta
PruningGomoku
Competition

Qiao Tan and Hu Xiaoti.

CS221 Project Final Report Gomoku Game Agent.

Technical report.



J H Kang and H J Kim.

Effective Monte-Carlo tree search strategies for Gomoku AI.

Int. J. Comput. Technol. Appl., 9:4833–4841, 2016.

Zhentao Tang, Dongbin Zhao, Kun Shao, and Le Lv.

ADP with MCTS algorithm for Gomoku.

2016 IEEE Symp. Ser. Comput. Intell. SSCI 2016, (61273136), 2017.

Dennis J.N.J. Soemers, Chiara F. Sironi, Torsten Schuster, and Mark H.M. Winands.

Enhancements for real-time Monte-Carlo Tree Search in General Video Game Playing.

In IEEE Conf. Comput. Intell. Games, CIG, 2017.Abdallah Saffidine, Tristan Cazenave, and Université Paris-dauphine.
Generalized Proof Number Search.
1994.

Chih-hung Chen, Shun-shii Lin, and Yen-chi Chen.

An Algorithmic Design and Implementation of Outer-Open Gomoku.

2017 2nd Int. Conf. Comput. Commun. Syst., pages 26–33, 2017.