Last-Branch and Speculative Pruning Algorithms for $\text{Max}^n$

Nathan Sturtevant
UCLA Computer Science Department*

*soon to be University of Alberta
Problem Overview
Problem Overview

- There have been notable successes in developing expert-level 2-player games
Problem Overview

- There have been notable successes in developing expert-level 2-player games
- We’d like to develop programs to play multi-player games well
Minimax
Minimax

• Most commonly used 2-player decision rule
Minimax

- Most commonly used 2-player decision rule
- Implemented with alpha-beta pruning
Minimax

- Most commonly used 2-player decision rule
- Implemented with alpha-beta pruning
  - In best case, alpha-beta reduces tree size from $b^d$ to $b^{d/2}$
Minimax

- Most commonly used 2-player decision rule
- Implemented with alpha-beta pruning
- In best case, alpha-beta reduces tree size from $b^d$ to $b^{d/2}$
- Approach best case in practice by ordering nodes well
Max^n
Generalization of minimax to $n$ players
Max$^n$

- Generalization of minimax to $n$ players
- Luckhardt and Irani, 1986
Max^n

- Generalization of minimax to n players
- Luckhardt and Irani, 1986
- Static evaluation returns n-tuple of scores
Generalization of minimax to $n$ players

- Luckhardt and Irani, 1986

- Static evaluation returns $n$-tuple of scores
- Each player tries to maximize their own component of the $n$-tuple
Max^n Decision Rule

1

2

3

(3, 5, 2)

2

3

(4, 3, 3)

2

3

(1, 3, 6)

2

3

(2, 6, 2)
Max^n Decision Rule

1

2

3

(3, 5, 2)

(4, 3, 3)

(1, 3, 6)

(2, 6, 2)
Max^n Decision Rule

(3, 5, 2)  (4, 3, 3)  (1, 3, 6)  (2, 6, 2)
Max^n Decision Rule
Max$^n$ Decision Rule

(3, 5, 2)  (4, 3, 3)  (1, 3, 6)  (2, 6, 2)
Max$^n$ Decision Rule
Max$^n$ Decision Rule

(3, 5, 2) (4, 3, 3) (1, 3, 6) (2, 6, 2)
Max^n Decision Rule
Max,$^n$ Decision Rule

(3, 5, 2)

1

(3, 5, 2) → 2

3
(3, 5, 2)

(4, 3, 3) → 3

(1, 3, 6)

(2, 6, 2) → 3

(2, 6, 2)
Max\(^n\) Decision Rule

![Decision Tree Diagram]
Outline

• $\text{Max}^n$ Decision Rule
• $\text{Max}^n$ Pruning Techniques
• Experimental Results
• Conclusions
Previous Max$^n$ Pruning
Previous $\text{Max}^n$ Pruning

- Shallow Pruning (Korf, 1991)
Previous Max$^n$ Pruning

- Shallow Pruning (Korf, 1991)
- Alpha-Beta Branch and Bound Pruning (Sturtevant and Korf, 2000)
Previous $\text{Max}^n$ Pruning

- Shallow Pruning (Korf, 1991)
- Alpha-Beta Branch and Bound Pruning (Sturtevant and Korf, 2000)
- Are not always effective and/or applicable
Previous $\text{Max}^n$ Pruning

- Shallow Pruning (Korf, 1991)
- Alpha-Beta Branch and Bound Pruning (Sturtevant and Korf, 2000)
- Are not always effective and/or applicable
  - Effectiveness depends on both node ordering and static evaluation function
Max$^n$ Pruning
Assume at least:

$\max^n$ Pruning
Max^n Pruning

- Assume at least:
- Lower bound on each score (0)
Max^n Pruning

• Assume at least:
  • Lower bound on each score (0)
  • Upper bound on sum of all scores \((\text{maxsum})\)
Max^n Pruning

• Assume at least:
  • Lower bound on each score (0)
  • Upper bound on sum of all scores ($maxsum$)
• If the sum of scores always equals $maxsum$, the game is constant-sum
Shallow Pruning

$maxsum = 10$
Shallow Pruning

$maxsum = 10$

(5, 4, 1) (5, 3, 2)
maxsum = 10

Shallow Pruning
Shallow Pruning

maxsum = 10

(5, 4, 1) (≥5, 1)

(5, 4, 1) (5, 3, 2)
Shallow Pruning

$maxsum = 10$

($\geq 5, \leq 5, \leq 5$)

1

($5, 4, 1$)

2

($5, 4, 1$)  ($5, 3, 2$)

3

3
Shallow Pruning

$maxsum = 10$

(5, 4, 1) ≤ (5, 3, 2) ≤ (0, 6, 4)

(5, 4, 1) (≥5, ≤5, ≤5)
Shallow Pruning

\[ \text{maxsum} = 10 \]

\[
\begin{align*}
&1 \\
&\quad \rightarrow \begin{cases} 
(5, 4, 1) \\
(\geq 5, \leq 5, \leq 5) \\
(\leq 4, \geq 6, \leq 4) 
\end{cases} \\
&\quad \rightarrow \begin{cases} 
(5, 3, 2) \\
(0, 6, 4) 
\end{cases} \\
\end{align*}
\]
Shallow Pruning

$maxsum = 10$

(5, 4, 1) - (≥5, ≤5, ≤5)

(5, 3, 2) - (≤4, ≥6, ≤4)

(0, 6, 4)
Shallow Pruning

\[ \text{maxsum} = 10 \]

\[
\begin{align*}
(5, 4, 1) & \quad (\geq 5, \leq 5, \leq 5) \\
(5, 4, 1) & \quad (\leq 4, \geq 6, \leq 4) \\
\end{align*}
\]
Shallow Pruning

\[ \text{maxsum} = 10 \]
Shallow Pruning
Shallow Pruning

- Average case model predicts no asymptotic gain (Korf, 1991)
Shallow Pruning

- Average case model predicts no asymptotic gain (Korf, 1991)
- Dependent on properties of evaluation function
Shallow Pruning

- Average case model predicts no asymptotic gain (Korf, 1991)
- Dependent on properties of evaluation function
- For some games, natural evaluation functions allow no pruning
Deep Pruning
Deep Pruning

- Can we prune deeper in the tree based on the same bound?
Deep Pruning

- Can we prune deeper in the tree based on the same bound?
- Not directly valid in multi-player games
Deep Pruning

$maxsum = 10$
Deep Pruning

$maxsum = 10$

(4, 4, 2)
Deep Pruning

$maxsum = 10$

$(\geq 4, \leq 6, \leq 6)$

$(4, 4, 2)$
Deep Pruning

$maxsum = 10$

$(\geq 4, \leq 6, \leq 6)$

2

$(4, 4, 2)$

3

$(3, 2, 5)$
Deep Pruning

$maxsum = 10$

1

2

(≥4, ≤6, ≤6)

(4, 4, 2)

(≤8, ≥2, ≤8)

2

3

(3, 2, 5)
Deep Pruning

$maxsum = 10$

$(\geq 4, \leq 6, \leq 6)$

$(\leq 8, \geq 2, \leq 8)$

$(3, 2, 5)$

$(3, 1, 6)$
Deep Pruning

$maxsum = 10$

$\text{maxsum} = 10$

$\geq 4, \leq 6, \leq 6$
Deep Pruning

\[ \text{maxsum} = 10 \]

\[
\begin{align*}
& \geq 4, \leq 6, \leq 6 \\
& 4, 4, 2 \\
& 3, 2, 5 \\
& \leq 4, \leq 4, \geq 6 \\
& 3, 1, 6 \\
& \leq 4, \geq 2, \leq 8 \\
& \leq 8, \geq 2, \leq 8 \\
& 3, 4, 2
\end{align*}
\]
Deep Pruning

maxsum = 10

(≥4, ≤6, ≤6)

(4, 4, 2)

(≤8, ≥2, ≤8)

(3, 2, 5)

(≤4, ≤4, ≥6)

(3, 1, 6)
Deep Pruning

$maxsum = 10$

(≥4, ≤6, ≤6) → 1

(≤8, ≥2, ≤8) → 2

(≤4, ≤4, ≥6) → 3

(3, 1, 6) → 1

(0, 3, 7) → 1
Deep Pruning

\[ \text{maxsum} = 10 \]

- **1**
  - **2**
    - **3**
      - **1**
        - **2**
          - **3**
            - **1**
              - **0**, **3**, **7**

- **≥4, ≤6, ≤6**
- **≤8, ≥2, ≤8**
- **≤4, ≤4, ≥6**
- **(4, 4, 2)**
- **(3, 2, 5)**
- **(3, 1, 6)**
Deep Pruning

$\text{maxsum} = 10$

$1 \geq 4, \leq 6, \leq 6$}

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$

$2 \leq 8, \geq 2, \leq 8$

$3 \leq 4, \leq 4, \geq 6$

$1 \leq 4, \leq 4, \geq 6$
Deep Pruning

$maxsum = 10$

(≥4, ≤6, ≤6)

(≤8, ≥2, ≤8)

(≤4, ≤4, ≥6)

maxsum = 10

(3, 1, 6)

(0, 3, 7)
Deep Pruning

maxsum = 10

1

(≥4, ≤6, ≤6)

2

(≤8, ≥2, ≤8)

3

(≤4, ≤4, ≥6)

1

(3, 2, 5)

1

(3, 1, 6)

(0, 3, 7)
Deep Pruning

$maxsum = 10$

(\(\geq 4, \leq 6, \leq 6\))

(\(\leq 8, \geq 2, \leq 8\))

(\(\leq 4, \leq 4, \geq 6\))

(\(\leq 4, \leq 2, \leq 8\))

(\(\leq 4, \leq 4, \geq 6\))

(\(\geq 6, \leq 6, \leq 6\))
Deep Pruning

$maxsum = 10$

1

$(\geq 4, \leq 6, \leq 6)$

2

$(\leq 8, \geq 2, \leq 8)$

$(4, 4, 2)$

3

$(\leq 4, \leq 4, \geq 6)$

$(3, 2, 5)$

1

$(3, 1, 6)$

1

$(0, 3, 7)$

$\text{maxsum} = 10$
Deep Pruning

maxsum = 10

(≥4, ≤6, ≤6)

(≤8, ≥2, ≤8)

(≤4, ≤4, ≥6)

(3, 1, 6)

(0, 3, 7)
Deep Pruning

maxsum = 10

1

2

3

4, 4, 2

3, 2, 5

3

(≥4, ≤6, ≤6)

(≤8, ≥2, ≤8)

(≤4, ≤4, ≥6)

(0, 3, 7)

(6, 3, 1)
Deep Pruning

$maxsum = 10$

1

$\geq 4, \leq 6, \leq 6$

2

4, 4, 2

$\leq 8, \geq 2, \leq 8$

3

3, 2, 5

$\leq 4, \leq 4, \geq 6$

1

3, 1, 6

3

6, 3, 1

3

0, 3, 7
Deep Pruning

\( maxsum = 10 \)

\[
\begin{aligned}
\text{1} & \quad (\geq 4, \leq 6, \leq 6) \\
\text{2} & \quad (\leq 8, \geq 2, \leq 8) \\
\text{3} & \quad (\leq 4, \leq 4, \geq 6) \\
\text{1} & \quad (3, 1, 6) \\
\end{aligned}
\]
Deep Pruning
Deep Pruning

- The $\max^n$ value of the last child cannot be the $\max^n$ value of the tree
Deep Pruning

- The $\max^n$ value of the last child cannot be the $\max^n$ value of the tree
- It can still affect the $\max^n$ value of the tree (Korf, 1991)
Deep Pruning

• The $\max^n$ value of the last child cannot be the $\max^n$ value of the tree

• It can still affect the $\max^n$ value of the tree (Korf, 1991)

• Is there a valid way to do this pruning?
Last-Branch Pruning
(Part 1)

maxsum = 10
Last-Branch Pruning

(Part 1)

$maxsum = 10$

(maxsum = 10)

(≥4, …)  1  …

(4, 4, 2)  2  (, ≥2, )

(3, 2, 5)  3  (…, ≥6)

(3, 1, 6)  1

(0, 3, 7)  1
**Last-Branch Pruning**

(Part 1)

$maxsum = 10$

```
(3, 1, 6) -- 1  
(3, 2, 5) -- 3  
(4, 4, 2) -- 2  
(≥4, ...)    
(..., ≥6)    
(, ≥2, )     
...
```

(maxsum = 10)
Last-Branch Pruning (Part 1)

$maxsum = 10$

Diagram:
- Node 1: $(3, 1, 6)$
- Node 2: $(4, 4, 2)$
- Node 3: $(3, 2, 5)$
- Node 1: $(0, 3, 7)$

Highlights:
- $(3, 1, 6)$:
  - $(≥4, ...)$
- $(4, 4, 2)$:
  - $(, ≥2, )$
- $(3, 2, 5)$:
  - $(..., ≥6)$
- $(0, 3, 7)$
Last-Branch Pruning

(Part 1)

$maxsum = 10$

\begin{align*}
(4, 4, 2) & \geq 4, \ldots \\
(3, 2, 5) & \geq 6 \\
(0, 3, 7) & \\
\end{align*}
Last-Branch Pruning

(Part 1)

\[ \text{maxsum} = 10 \]
**Last-Branch Pruning**

*(Part 1)*

$maxsum = 10$

![Diagram](image)
Last-Branch Pruning
(Part 1)

\[ \text{maxsum} = 10 \]

\[
\begin{array}{c}
1 \\
(\geq 4, \ldots) \\
2 \\
(4, 4, 2) \\
3 \\
(3, 2, 5) \\
1 \\
(3, 1, 6)
\end{array}
\]

\[
\begin{array}{c}
2 \\
(\geq 2, \ldots) \\
3 \\
(\ldots, \geq 6) \\
1 \\
(0, 3, 7)
\end{array}
\]
Last-Branch Pruning I
Last-Branch Pruning I

- We can now deep prune when:
Last-Branch Pruning I

- We can now deep prune when:
- Scores sum to $\text{maxsum}$ or greater
Last-Branch Pruning I

- We can now deep prune when:
  - Scores sum to $\text{maxsum}$ or greater
  - Second player on their last branch
Last-Branch Pruning 1

- We can now deep prune when:
  - Scores sum to \textit{maxsum} or greater
  - Second player on their last branch
- Can we still do better?
Last-Branch Pruning (Part 2)

\[ maxsum = 10 \]

1

(4, 4, 2)

2

(3, 2, 5)

3

(3, 1, 6)

(≥4, …)

(, ≥2, )

(…, ≥6)
Last-Branch Pruning
(Part 2)

maxsum = 10

(≥4, ...)

(4, 4, 2)

(, ≥2, )

(3, 2, 5)

1
Last-Branch Pruning
(Part 2)

$\text{maxsum} = 10$

$\begin{align*}
(\geq 4, \ldots) & \quad 1 \\
(4, 4, 2) & \quad 2 \\
(3, 2, 5) & \quad 3 \\
(3, 3, 4) & \quad 1
\end{align*}$
Last-Branch Pruning
(Part 2)

\[ \text{maxsum} = 10 \]

The diagram illustrates the pruning process with nodes labeled from 1 to 3 and values in parentheses. The pruning criterion is based on the sum of values and the condition that any branch with a sum of 4 or more is pruned.
Last-Branch Pruning
Last-Branch Pruning

- Directional Algorithm
Last-Branch Pruning

- Directional Algorithm
- Prunes when:
Last-Branch Pruning

- Directional Algorithm
- Prunes when:
  - Sum of players’ bounds \( \geq \maxsum \)
Last-Branch Pruning

- Directional Algorithm
- Prunes when:
  - Sum of players’ bounds $\geq \text{maxsum}$
  - Second player on their last branch
Last-Branch Pruning

- Directional Algorithm
- Prunes when:
  - Sum of players’ bounds $\geq maxsum$
  - Second player on their last branch
- Limited to last branch! Can we do better?
Speculative Pruning

$maxsum = 10$

$\text{(4, 4, 2)}$ (≥4, …)

$\text{(3, 2, 5)}$ (…, ≥4)

$\text{(3, 3, 4)}$ (2, 3, 5)
Speculative Pruning

maxsum = 10

1
(≥4, …)

2
(4, 4, 2)

(≥2, )

3
(3, 2, 5)

(…, ≥4)

1
(3, 3, 4)

…

2

3

(6, 3, 1)

1
(2, 3, 5)
Speculative Pruning

$\text{maxsum} = 10$

$\begin{array}{ccc}
(4, 4, 2) & \geq 4, \ldots & \\
(3, 2, 5) & \ldots, \geq 4 & \\
(3, 3, 4) & & (2, 3, 5)
\end{array}$
Speculative Pruning

\[ \text{maxsum} = 10 \]
Speculative Pruning

\[ \text{maxsum} = 10 \]

\[
\begin{array}{c}
\text{1} \\
\text{2} \\
\text{3} \\
\text{1}
\end{array}
\]

\[
\begin{array}{c}
\text{2} \\
\text{3} \\
\text{1}
\end{array}
\]

\[
\begin{array}{c}
\text{3} \\
\text{1}
\end{array}
\]

\[
\begin{array}{c}
\text{1}
\end{array}
\]

\[
\begin{array}{c}
\text{4, 4, 2} \\
\text{3, 2, 5} \\
\text{3, 3, 4}
\end{array}
\]

\[
\begin{array}{c}
\text{≥4, …} \\
\text{≥2, } \\
\text{≥4}
\end{array}
\]

\[
\begin{array}{c}
\text{6, 3, 1} \\
\text{2, 3, 5}
\end{array}
\]
Speculative Pruning

$maxsum = 10$

![Diagram of Speculative Pruning](image)
Speculative Pruning

$maxsum = 10$

1

2

4, 4, 2 (≥4, …)

3

3, 2, 5 (…, ≥4)

1

3, 3, 4

2

, ≥2, )

3

3

3

2

1
Speculative Pruning

$maxsum = 10$

1

2

$(4, 4, 2)$

3

$(3, 2, 5)$

1

$(3, 3, 4)$

$(\geq 4, \ldots)$

$(\ , \geq 2, \ )$

$(\ldots, \geq 4)$

$\ldots$
Speculative Pruning

$maxsum = 10$

(≥4, …)

(4, 4, 2)

(3, 2, 5)

(3, 3, 4)

2

(≥2, )

3

(…, ≥4)

1

3
Speculative Pruning

$maxsum = 10$

\[\left(4, 4, 2\right), \left(3, 2, 5\right), \left(3, 3, 4\right)\]

\[\left(\geq 4, \ldots\right), \left(\ldots, \geq 4\right)\]
Speculative Pruning

$maxsum = 10$

1

≥4, ...

2

4, 4, 2

(4, 4, 2)

≥2

(≥2, )

3

3, 2, 5

(3, 2, 5)

≥4

(…, ≥4)

3

6, 3, 1

Oops!

re-search
Speculative Pruning

maxsum = 10
Speculative Pruning

$maxsum = 10$

![Tree Diagram]

- Node 1: $(\geq 4, \ldots)$
- Node 2: $(4, 4, 2)$
- Node 3: $(3, 2, 5)$
- Node 1: $(3, 3, 4)$
Speculative Pruning

maxsum = 10

(≥4, …)

(, ≥2, )

(…, ≥4)

(4, 3, 3)
Speculative Pruning

maxsum = 10

No Problem!
Speculative Pruning

\[ \text{maxsum} = 10 \]

1. \((≥4, …)\)
   - 2. \((4, 4, 2)\) (, \(≥2, \))
     - 3. \((3, 2, 5)\) (…, \(≥4\))
       - 1. \((3, 3, 4)\)
       - 2. \((3, 3, 4)\)
     - 3. \((4, 4, 2)\)
   - 2. \((3, 3, 4)\)
     - 3. \((3, 3, 4)\)
   - 3. \((4, 3, 3)\)

...
Speculative Pruning
Speculative Pruning

- Non-directional algorithm
Speculative Pruning

- Non-directional algorithm
- Prunes are not guaranteed to be correct
Speculative Pruning

- Non-directional algorithm
- Prunes are not guaranteed to be correct
- Can re-search branches if needed
Speculative Pruning

- Non-directional algorithm
- Prunes are not guaranteed to be correct
- Can re-search branches if needed
- Works on any constant-sum game tree
Speculative Pruning

- Non-directional algorithm
- Prunes are not guaranteed to be correct
- Can re-search branches if needed
- Works on any constant-sum game tree
- Effectiveness depends only on node ordering
Speculative Pruning
Speculative Pruning

- Reduces branching factor:
Speculative Pruning

- Reduces branching factor:
- In the best case, as $b$ gets large
Speculative Pruning

- Reduces branching factor:
  - In the best case, as $b$ gets large
  - $b^{(n-1)/n}$
Speculative Pruning

- Reduces branching factor:
  - In the best case, as $b$ gets large
  - $b^{(n-1)/n}$
- Effective even in the average case
## Speculative Pruning
### (3-Players)

<table>
<thead>
<tr>
<th>b</th>
<th>$b^{2/3}$</th>
<th>actual b</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.59</td>
<td>1.84</td>
</tr>
<tr>
<td>3</td>
<td>2.08</td>
<td>2.47</td>
</tr>
<tr>
<td>4</td>
<td>2.52</td>
<td>3.00</td>
</tr>
<tr>
<td>10</td>
<td>4.64</td>
<td>5.42</td>
</tr>
<tr>
<td>1000</td>
<td>100.0</td>
<td>103.6</td>
</tr>
</tbody>
</table>
Outline

• $\text{Max}^n$ Decision Rule
• $\text{Max}^n$ Pruning Techniques
• Experimental Results
• Conclusions
Chinese Checkers
Chinese Checkers
Chinese Checkers

• 3 Players
Chinese Checkers

- 3 Players
- Played 30 games, ~1500 searches
Chinese Checkers

- 3 Players
- Played 30 games, ~1500 searches
- Iterative deepening search
Chinese Checkers

- 3 Players
- Played 30 games, ~1500 searches
  - Iterative deepening search
  - Limited branching factor to 10 moves
Chinese Checkers

- 3 Players
- Played 30 games, ~1500 searches
  - Iterative deepening search
  - Limited branching factor to 10 moves
  - Measured average expansions at depth 6
Chinese Checkers

- 3 Players
- Played 30 games, ~1500 searches
  - Iterative deepening search
  - Limited branching factor to 10 moves
- Measured average expansions at depth 6
- Previous algorithms could not prune tree
# Chinese Checkers

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Node expansions (depth 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max(^n)</td>
<td>1.2 million</td>
</tr>
<tr>
<td>Spec. Max(^n)</td>
<td>100k</td>
</tr>
</tbody>
</table>
Hearts and Spades
Hearts and Spades

- Trick-based card games
Hearts and Spades

- Trick-based card games
- Have monotonic heuristics
Hearts and Spades

- Trick-based card games
- Have monotonic heuristics
- Shallow pruning does not occur in Hearts
Hearts and Spades

• Trick-based card games
• Have monotonic heuristics
• Shallow pruning does not occur in Hearts
• Measured average search depth
Hearts and Spades

- Trick-based card games
- Have monotonic heuristics
- Shallow pruning does not occur in Hearts
- Measured average search depth
- Iterative deepening search
Hearts and Spades

- Trick-based card games
- Have monotonic heuristics
- Shallow pruning does not occur in Hearts
- Measured average search depth
- Iterative deepening search
- 500k node search limit
## Average Search Depth

<table>
<thead>
<tr>
<th></th>
<th>Hearts</th>
<th>Spades</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best Prev. Max^n</strong></td>
<td>20.9</td>
<td>21.7</td>
</tr>
<tr>
<td><strong>Spec. Max^n</strong></td>
<td>22.1</td>
<td>24.3</td>
</tr>
</tbody>
</table>
Conclusions
Conclusions

• New pruning algorithms for $\text{max}^n$
Conclusions

- New pruning algorithms for $\text{max}^n$
- Last-Branch Pruning
Conclusions

• New pruning algorithms for $\max^n$
• Last-Branch Pruning
• Directional algorithm
Conclusions

• New pruning algorithms for $\text{max}^n$
• Last-Branch Pruning
• Directional algorithm
• Speculative Pruning
Conclusions

• New pruning algorithms for $\max^n$
• Last-Branch Pruning
  • Directional algorithm
• Speculative Pruning
  • Non-directional algorithm
Conclusions

• New pruning algorithms for $\text{max}^n$
• Last-Branch Pruning
  • Directional algorithm
• Speculative Pruning
  • Non-directional algorithm
• First algorithms effective in pruning any constant-sum game