Exhaustive and Semi-Exhaustive Procedural Content Generation

Nathan R. Sturtevant, University of Alberta
Matheus Jun Ota, University of Campinas

AIIDE
November 15, 2018
PCG Taxonomy
PCG Taxonomy

- Constructive, grammars, etc.
PCG Taxonomy

- Constructive, grammars, etc.
- Search-Based PCG
PCG Taxonomy

- Constructive, grammars, etc.
- Search-Based PCG
  - Evolutionary algorithms
PCG Taxonomy

• Constructive, grammars, etc.
• Search-Based PCG
  • Evolutionary algorithms
  • Other approaches
PCG Taxonomy

• Constructive, grammars, etc.
• Search-Based PCG
  • Evolutionary algorithms
  • Other approaches
  • Exhaustive search
PCG Taxonomy

- Constructive, grammars, etc.
- Search-Based PCG
  - Evolutionary algorithms
  - Other approaches
    - Exhaustive search
    - Random search
PCG Taxonomy

- Constructive, grammars, etc.
- Search-Based PCG
  - Evolutionary algorithms
  - Other approaches
    - Exhaustive search
    - Random search
    - Solver-based (eg ASP)
Exhaustive PCG
Exhaustive PCG

• Synthesize work from other fields (e.g. mathematics) as reference for EPCG approaches
Exhaustive PCG

• Synthesize work from other fields (e.g. mathematics) as reference for EPCG approaches
• Give samples of underlying algorithms for EPCG
Exhaustive PCG

• Synthesize work from other fields (e.g. mathematics) as reference for EPCG approaches
• Give samples of underlying algorithms for EPCG
• Examples of the use of EPCG
  • Mixed-initiative process
The Witness

Spoiler Alert!
The Design Goal

• Solve 3 puzzles simultaneously with the same path
• How do we choose three puzzles?
  • Secret sharing algorithm
Are you sure this can be solved?
EPCG
EPCG

• **Exhaustive PCG** describes approaches for generating procedural content where all possible content is *methodically generated* and evaluated.
EPCG

• **Exhaustive PCG** describes approaches for generating procedural content where all possible content is *methodically generated* and evaluated.

• Algorithms that are capable of methodically generating all content, but that choose to skip some content are *semi-exhaustive*. 
EPCG
EPCG

• Evaluator
  • Evaluates the utility of a given state
EPCG

• **Evaluator**
  • Evaluates the utility of a given state

• **Generator**
  • Enumerates all possible states
    • Combinations
    • Permutations
    • Multi-set
  • Can also be done recursively on variables/values
General Operations
General Operations

- $\text{maxRank}$
  - Comes from computing the total number of configurations
General Operations

- `maxRank`
  - Comes from computing the total number of configurations
- `hash ← Rank(s)`
  - Perfect hash function for a state
General Operations

• maxRank
  • Comes from computing the total number of configurations
• hash ← Rank(s)
  • Perfect hash function for a state
• s ← Unrank(hash)
  • Converts a hash back into a state
Example 1
Board
Board

16 locations
3 pieces
11 piece types
Board

Library

16 locations
3 pieces
11 piece types
16 locations
3 pieces
11 piece types

\[
\binom{16}{3} \times 11^3 = 745,360
\]
Exhaustive Approach
Exhaustive Approach

For $i = 0 \rightarrow maxRank$
Exhaustive Approach

\[
\text{For } i = 0 \rightarrow \text{maxRank} \\
\text{s } \leftarrow \text{Unrank}(i)
\]
Exhaustive Approach

For $i = 0 \rightarrow maxRank$

$s \leftarrow \text{Unrank}(i)$

localEval $\leftarrow \text{Eval}(s)$
Exhaustive Approach

For $i = 0 \rightarrow maxRank$

$s \leftarrow \text{Unrank}(i)$

localEval $\leftarrow \text{Eval}(s)$

if (localEval > globalEval)
Exhaustive Approach

For $i = 0 \rightarrow maxRank$

$s \leftarrow \text{Unrank}(i)$

localEval $\leftarrow \text{Eval}(s)$

if (localEval $>$ globalEval)
    globalEval $= \text{localEval}$
Exhaustive Approach

For \( i = 0 \rightarrow \text{maxRank} \)

\[
\begin{align*}
    &s \leftarrow \text{Unrank}(i) \\
    &\text{localEval} \leftarrow \text{Eval}(s) \\
    &\text{if } (\text{localEval} > \text{globalEval}) \\
    &\hspace{1em} \text{globalEval} = \text{localEval} \\
    &\hspace{1em} \text{best} = i
\end{align*}
\]
Virtual location of each piece is as far as possible from actual piece.
Virtual location of each piece is as far as possible from actual piece
Virtual location of each piece is as far as possible from actual piece.
Virtual location of each piece is as far as possible from actual piece.
Example 2
Board
Board

9 locations
3 pieces
12 piece types
Board

9 locations
3 pieces
12 piece types

\[
\binom{9}{3} 12^3 = 145,152
\]
Board

9 locations
3 pieces
12 piece types

\[
\binom{9}{3} 12^3 = 145,152
\]

Triples: $3.06 \times 10^{15}$
Board

9 locations
3 pieces
12 piece types

\[ \binom{9}{3} 12^3 = 145,152 \]

Triples: \(3.06 \times 10^{15}\)

Semi-Exhaustive: Branch and Bound to prune suboptimal solutions
Semi-Exhaustive Approach
Semi-Exhaustive Approach

• Prune *single* boards with too few solutions
Semi-Exhaustive Approach

• Prune **single** boards with too few solutions
• Prune **pairs** of boards with too few solutions
Semi-Exhaustive Approach

• Prune single boards with too few solutions
• Prune pairs of boards with too few solutions
• Exhaustively enumerate remaining combinations
Semi-Exhaustive Approach

- Prune **single** boards with too few solutions
- Prune **pairs** of boards with too few solutions
- Exhaustively enumerate remaining combinations
  - Continue to prune combinations which are worse than best found so far
Semi-Exhaustive Approach

• Prune **single** boards with too few solutions
• Prune **pairs** of boards with too few solutions
• Exhaustively enumerate remaining combinations
  • Continue to prune combinations which are worse than best found so far

https://movingai.com/witness.html
Example 3
Fling!

- $\binom{56}{10} = 35.6$ billion boards with 10 pieces

- 15 million boards (0.04%) with 1 solution

- Forward search is expensive

- Not amenable to genetic operations
Retrograde Analysis

• Solve all boards, iteratively increasing the number of pieces on the board
  • 1 piece, 2 pieces, 3 pieces, etc
  • Easy to solve those with $n$ pieces by computing from those with $n-1$ pieces
• Avoid re-searching the underlying tree
• Requires the ranking function to look up states in memory
Retrograde Analysis
Retrograde Analysis

• For $i = 0 \rightarrow \text{maxRank}$
Retrograde Analysis

• For $i = 0 \rightarrow \text{maxRank}$
  • $s \leftarrow \text{Unrank}(i)$
Retrograde Analysis

• For $i = 0 \rightarrow \text{maxRank}$
  
  • $s \leftarrow \text{Unrank}(i)$
  
  • for each successor $s_i$ of parent
Retrograde Analysis

• For \( i = 0 \rightarrow \text{maxRank} \)
  
  • \( s \leftarrow \text{Unrank}(i) \)
  
  • for each successor \( s_i \) of parent
    
    • \( r \leftarrow \text{Rank}(s_i) \)
Retrograde Analysis

• For $i = 0 \rightarrow \text{maxRank}$
  • $s \leftarrow \text{Unrank}(i)$
  • for each successor $s_i$ of parent
    • $r \leftarrow \text{Rank}(s_i)$
    • Check if solvable/single solution
Retrograde Analysis

• For $i = 0 \rightarrow \text{maxRank}$
  • $s \leftarrow \text{Unrank}(i)$
  • for each successor $s_i$ of parent
    • $r \leftarrow \text{Rank}(s_i)$
      • Check if solvable/single solution
  • If solvable/single solution
Retrograde Analysis

• For \( i = 0 \rightarrow \text{maxRank} \)
  
  • \( s \leftarrow \text{Unrank}(i) \)
  
  • for each successor \( s_i \) of parent
    
    • \( r \leftarrow \text{Rank}(s_i) \)
    
    • Check if solvable/single solution
    
  • If solvable/single solution
    
    • mark \( i \) as single solution / solvable
Then...
Then...

• After efficiently identifying single solution puzzles
Then…

• After efficiently identifying single solution puzzles

• …run EPCG on these puzzles to choose the best
Takeaways
Takeaways

• Many puzzle problems are easy to analyze by computer - even if they are combinatorially large
Takeaways

• Many puzzle problems are easy to analyze by computer - even if they are combinatorially large

• EPCG enables us to ask precise questions about the space of possible puzzles
Takeaways

• Many puzzle problems are easy to analyze by computer - even if they are combinatorially large
• EPCG enables us to ask precise questions about the space of possible puzzles
• See the paper for some of the mathematics behind this analysis
Takeaways

• Many puzzle problems are easy to analyze by computer - even if they are combinatorially large
• EPCG enables us to ask precise questions about the space of possible puzzles
• See the paper for some of the mathematics behind this analysis
• Sample code on www.movingai.com