Soft Decomposition Search in Computer Go

Author: Keh-Hsun Chen

Presenter: Philip Henderson

Overview

- Past Uses of Decomposition Search
- Soft Decomposition Representation
- Resulting Properties
- Heuristic Evaluation
- Future Work

Decomposition Search

- Mathematical basis: Combinatorial Game Theory
- Can solve endgames far more efficiently (e.g. Müller 1999)
- Can use to solve one-eye tsumego problems (e.g. Kishimoto and Müller 2005)
- Goal of this paper is to apply it to the early and middle stages of a go game

Endgames

- Can solve global position optimally using local searches
- Greatly reduces the size of the search space

Limitations:

- Cyclic subgames (ko) can be problematic
- Intermediate combinatorial game expressions can become too complex

One-Eye Tsumego

- Search guided by df-pn search
- During search, eye-space region may become divided
- Can dynamically decompose to improve performance
- Solution returned is provably correct
- Limitation: ko in a sub-region does not interact with others

Relaxed Decomposition

- Not fully separated by safe stones, but they have a miai connection
- Idea is to allow for earlier and more decompositions
- Still correct if finds eye, but completeness unproven/unknown
- Cyclic sub-regions are still problematic

Soft Decomposition

- Previous uses are provably correct
- Want to apply decomposition earlier, as a heuristic instead
- Regions are not truly separated, but classified as components/sub-games

Sub-Games

- Two types: urgent and calm
- Urgent contains one or more unsafe groups
- Unsafe groups combined by transitive closure
- Safe groups form calm subgames
- Requires tool to make this decomposition dynamically during the game

Binary Game Trees

- BGTs are combinatorial game trees
- Restricted so that non-terminal nodes have two successors (one per player)
- Terminal positions assigned integer value (positive Left/Black, negative Right/White)
- Sum of these BGTs is a BGF: model of entire game position

More on BGTs

- $L_v(T)$ denotes value if alternate turns, and Left moves first
- Require local engine to select unique best moves for Left and Right
- BGTs omitted if clearly irrelevant/small
- BGT representation will not go to full local depth

Definitions

- $L_{choice}(k)$ is board state given that Left plays move in subgame k
- F^L is board state after optimal move by Left
- L_0 is end score given optimal play starting with Left
- Similar terms for Right player

Ordering BGFs

- Partial ordering $\geq$
- Requires that both the Left and Right optimal results are $\geq$
- Reflexive, transitive, anti-symmetric
- Integer k is terminal position
- Relation holds if add identical sub-games to each side

Inner/Outer Swings

- $L_v^\infty(T)$ denotes value if only Left plays
- Inner swing of T is $[L_v(T)|R_v(T)]$
- Outer swing of T is $[L_v^\infty(T)|R_v^\infty(T)]$

Dominance Relation

- $T_1 \gg T_2$ means Left and Right both prefer to play in T_1, regardless of the BGF
- Two games may be equally valuable, yet not equal
- Thus reflexive, transitive but not anti-symmetric
- $T \gg$ inner swing of T is generally true
- Outer swing of $T \gg$ inner swing of T

Incentives

- Basic value of move is size of inner swing of that component
- L-sente value of a move is $LL_v(T) - L_v(T)$
- A move is L-sente if its L-sente value $\geq$ basic value of any other move

Computation

For a BGT, the following can be computed in $O(h)$ time:

- Inner and outer swings
- L-sente and R-sente moves and values

A heuristic based on these values can guide the global move selection.

Possible Heuristics

1. Sente moves before gote, order by basic values (higher ones first)
2. Use basic value plus half the L-sente value as priority
3. Use basic value plus half L-sente value plus one quarter R-sente value as priority

Evaluation

- Terminal regions already decided - simply sum
- Regions sente for both get split in decreasing order ($\gg$) of inner swings
- Left assigned all exclusively L-sente regions, and likewise Right gets all R-sente ones
- Gote regions get split in decreasing order of inner swings

Search for Best Move

- Normal alpha-beta until reach depth limit
- Then use heuristic evaluation defined above
- Can compute values on which it is based efficiently

Conclusions

- Decomposition search previously used only when mathematically correct
- Current work attempts to use this concept as a heuristic for move ordering and evaluation
- Mathematical relations and properties can be used to guide global decisions

Future Work

- Implementation and experimental results still required
- Strong local go engine necessary (sub-game partitioning and best move selection)
- Can error bounds be placed on the performance of this heuristic?

Any Questions?