Playing it Safe: Recognizing Secure Territories in Computer Go

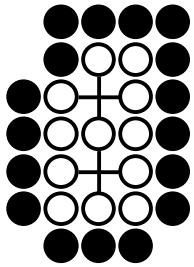
by Using Static Rules and Search

Martin Müller
University of Alberta

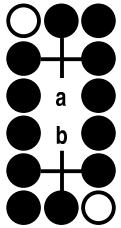
- Safety of Stones and Territory
- Recognizing Sure Liberties
- Static Methods
- Search for Sure Liberties

Safety of Stones - Normal Cases

- Two eyes: 2 x 1 sure liberty

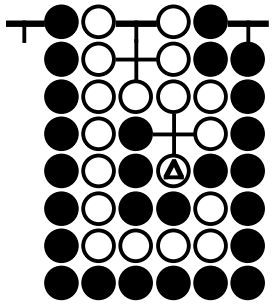


- Large area: 1 x 2 sure liberties

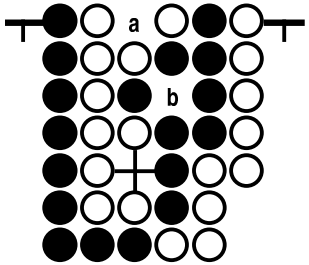


Safety of Stones - Other Cases

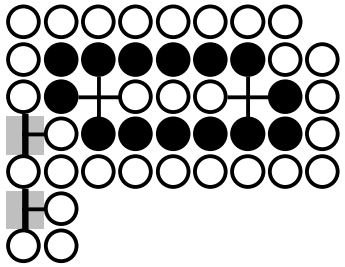
- Snapback: in *atari*, but safe



- *Ko:* no sure liberties, but safe



- Seki: safe by *shared* untouchable liberties

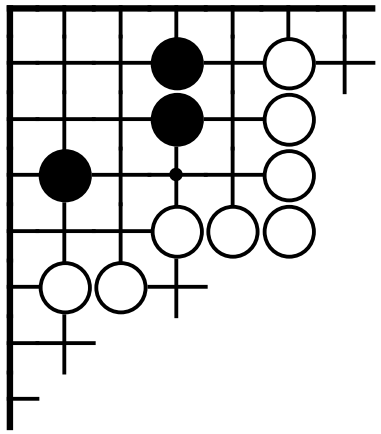


Static Evaluation vs. Search

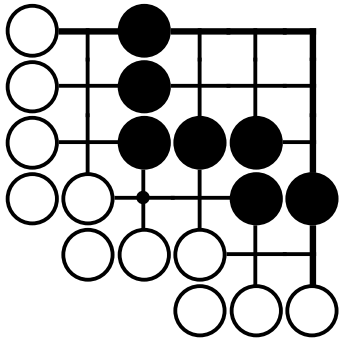
- Prove safety: static evaluation not enough
- Difficult cases must use search
- Good static evaluation can greatly speed up search
- Static analysis: decompose search into independent subtasks
- Aim: exponential speedup

Life&Death vs Safety Proofs

- can play on *outside*, create new eye space
- often uses *heuristics*
- *earlier* in the game

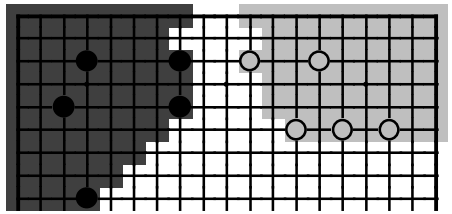


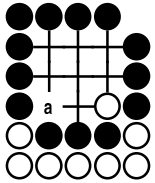
- restricted to *interior* of single region
- must be *exact*
- *later* in the game

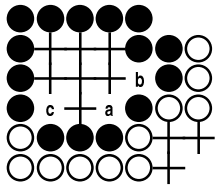


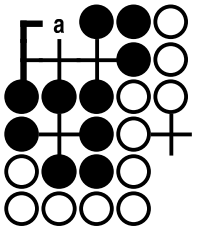
Safety Estimation in Current Go Programs

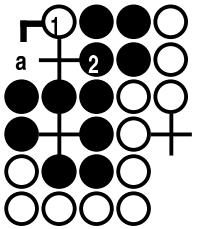
- Heuristics, Influence function
- Single-target capture search
- Multi-target capture search
- No *seki* recognition (only as capture)







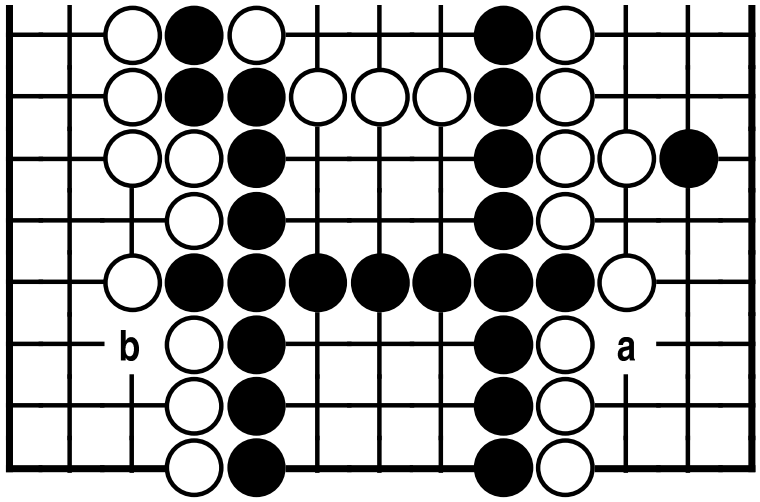




Handling of Unclear Situations

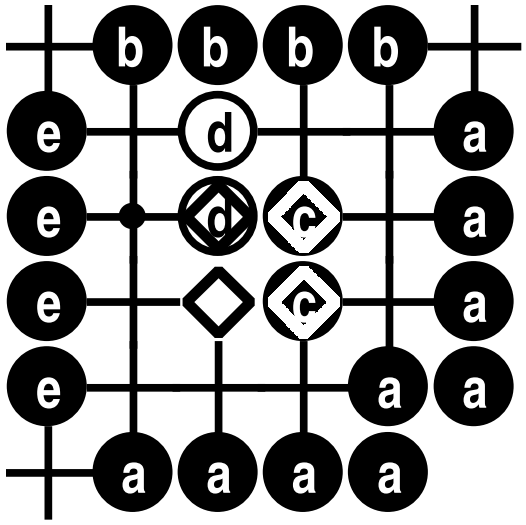
	Defend	Ignore
Danger	+	loss: BIG!
Safe	loss: 1 pt, sente	+

Most programs
defend both a and b



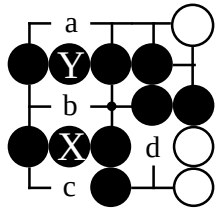
Blocks and Regions

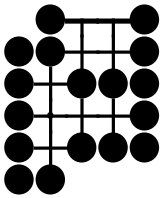
- Block:
  Unit of capture
- Region: surrounded by one player
- Enclosing blocks:
  a,b,e
- Interior: marked ◇

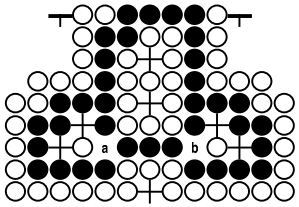


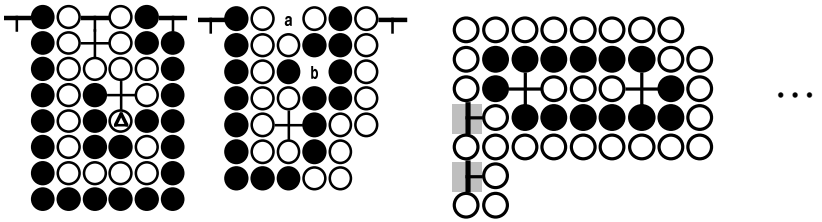
Types of Safety

- Unconditional safety (Benson)
- Locally alternating play
- Nonlocal safety by two sure liberties
- Other Safety









Sure Liberties and Safety

- Blocks B *alive* in regions R:

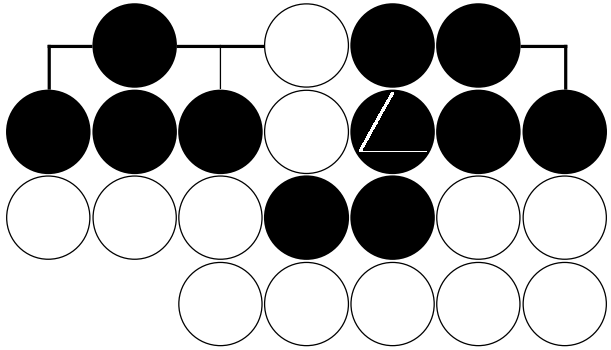
$$\forall b \in B \sum_{r \in R} SLC(b, r, B) \geq 2$$

- Benson: SLC(b,r,B) = 1 if r "healthy" for b
- Alternating play: SLC(b,r,B) = 1 or 2 recognize by
  - static rules
  - search

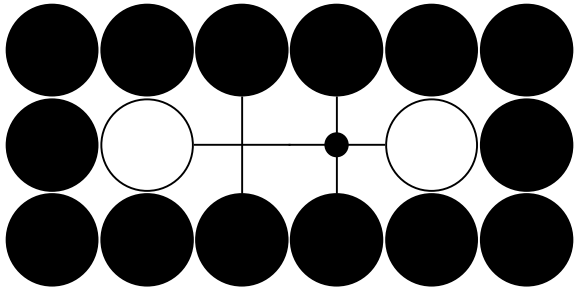
Sure Liberty Count *SLC* for Alternating Play

SLC(b, r, B) = *k*, *k* = 1 or 2:

- b is enclosing block of r
- Defender has strategy:
  - Either *all* blocks enclosing r have at least *k* liberties in r,
  - Or *some* blocks have only *k*-*1* liberties, but it is *defender's* turn



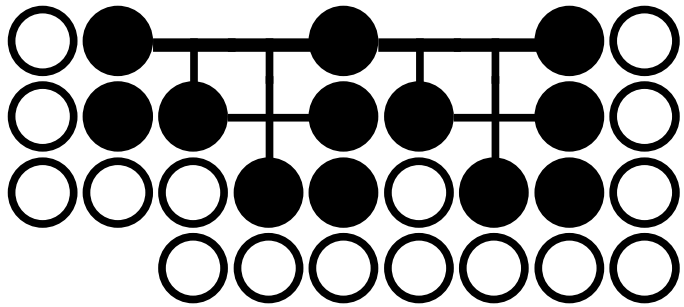
SLC = 1 (!)
if defender's turn



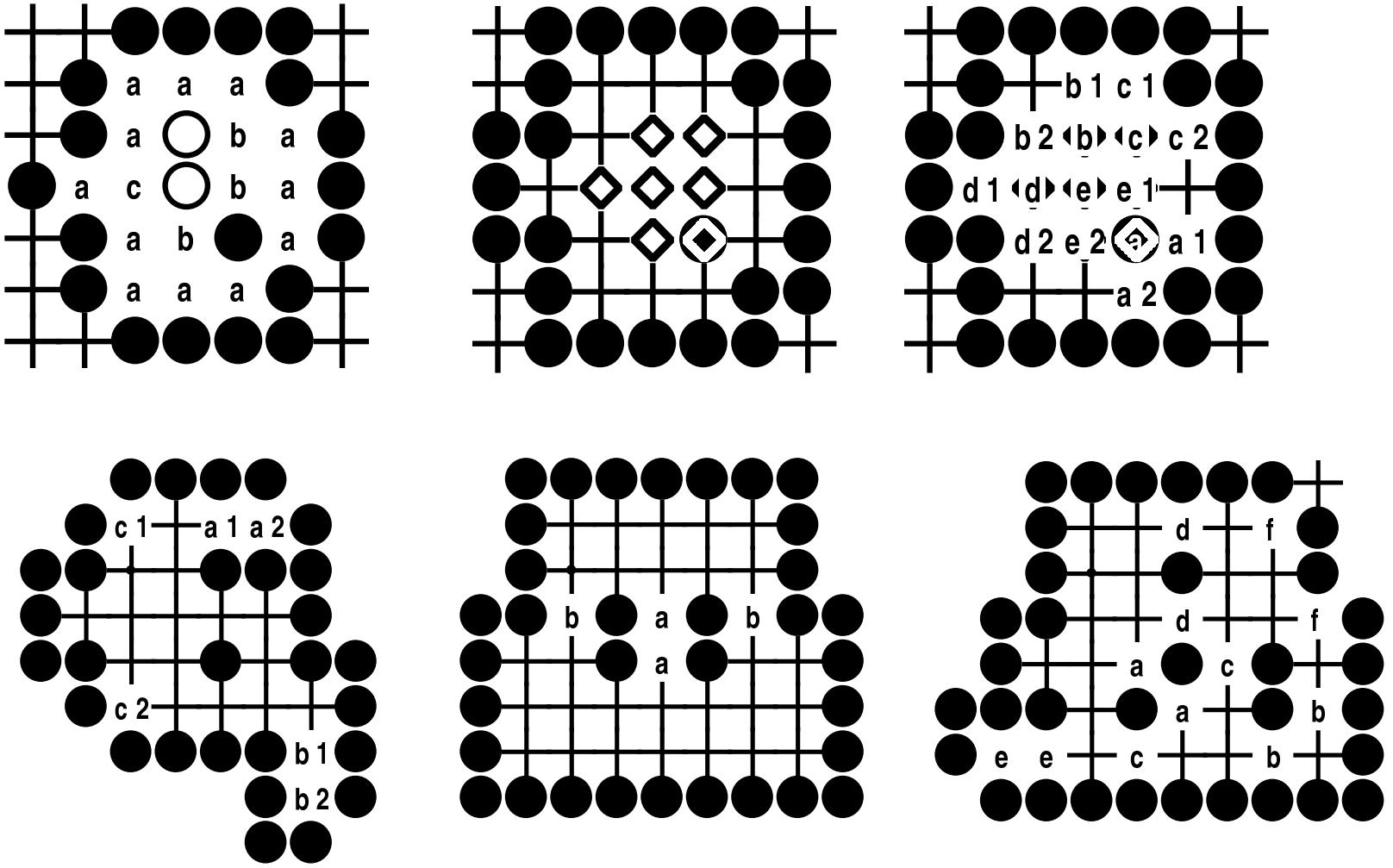
SLC = 2

Extension to Chains

- Blocks have only one sure liberty directly
- Second liberty: connect to another block
- Chain: set of surely connectible blocks
- SLC for *chains:* keep both liberty and connectivity

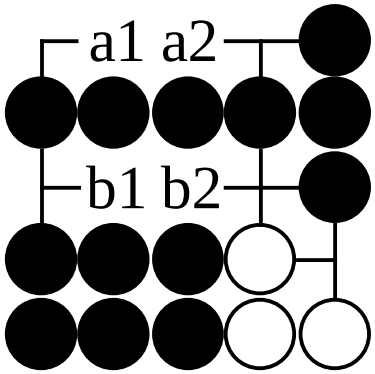


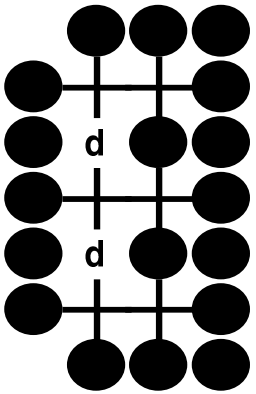
Static Recognition of 1 Sure Liberty



Static Recognition of 2 Sure Liberties

- All empty points accessible
- Two *intersection points*: can be split into two eyes





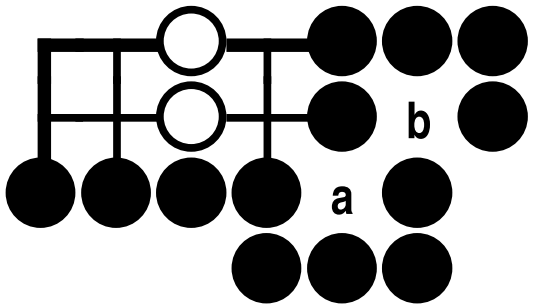
Search for Sure Liberties

- Standard alphabeta search
- Attacker moves first,
  wins all ko fights
- All legal local moves, plus Pass
- Evaluation:
  success, failure, unknown
- Static recognition in leaves

Region Surrounded by Safe Blocks

Surrounding blocks safe:

- Region need not provide liberties
- To prove: opponent cannot live inside
- Static Recognition:
  - No 2 eye points
  - *Nakade* shape



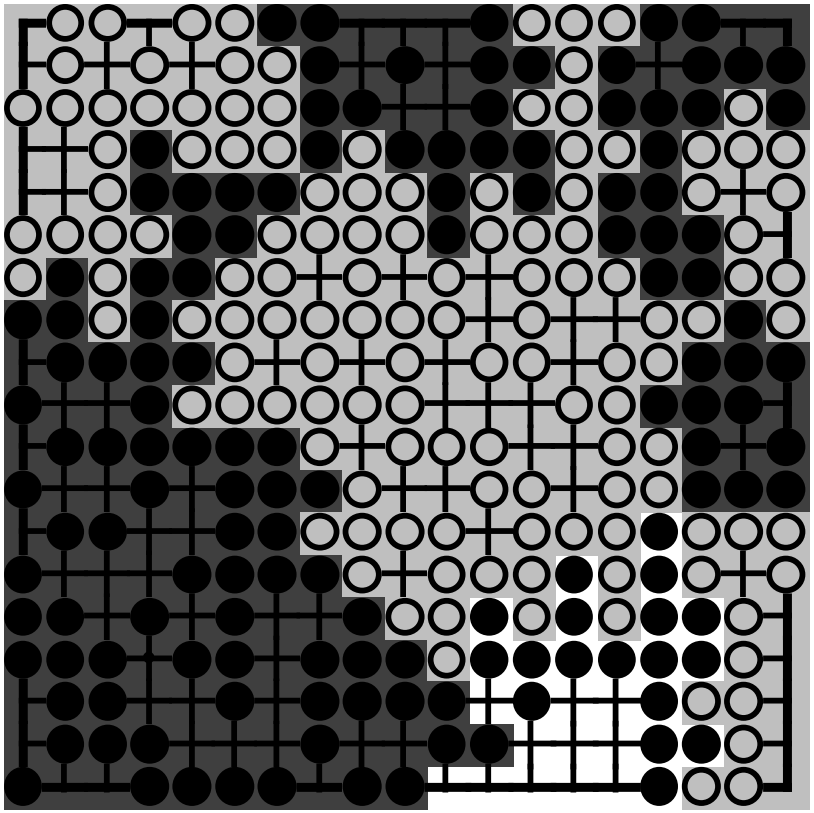
- Search: similar to sure liberty search

Test on Real Game Positions

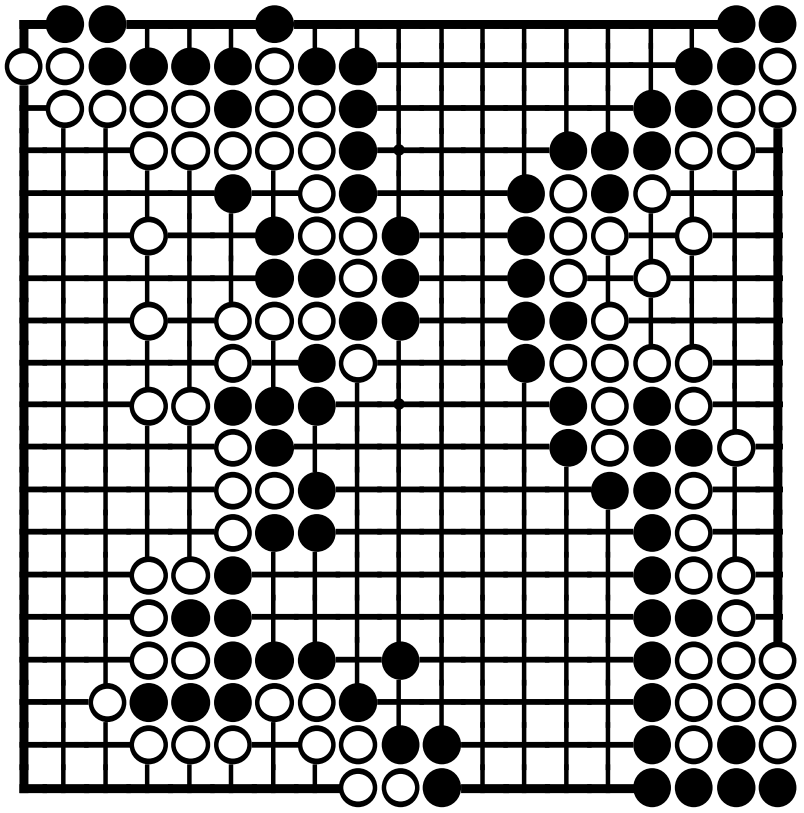
- Problem set: *IGS_31_counted* from Computer Go Test Collection
- Final positions of 31 games
- Search used:
  - Depth limit 6 ply
  - Three kinds:
    - recognize 1 sure liberty
    - recognize 2 sure liberties
    - regions surrounded by safe stones

Results

Method	Points	Blocks	Dame
Benson	1886 (16%)	103 (9%)	16
Static	2481 (22%)	168 (14%)	22
Search (1997)	2954 (26%)	198 (17%)	24
Search (2002)	about 40% (10 ply search(?))		



Best example in 1997:
92% classified
(in 2002: 100%)



Worst: 0% (5 of 31 games)

Applications

- Exact score counting
- Determine neccessary defensive moves
- Find *ko* threats: try move against safe territory, check if still safe
- Board partition: solve endgames by combinatorial game methods
- Recognize seki

Summary

- Prove Safety: Go subproblem,
  exact solution often possible
- Stones and territories handled differently
- New: better static rules and use of search
- Approach: local search for sure liberties

Future Work

Handle larger areas:

- Better partition plans
- Heuristic move ordering for search
- Better static tests for sure liberties

Other possible improvements:

- Generalize for non-closed regions
- Use outside liberties and connections
- Add *semeai* knowledge