Information Sharing in MCTS

UEC 7th Symposium 2013

Petr Baudiš ⟨pasky@ucw.cz⟩

Charles University

March 2013

Outline

1. Information Sharing
2. Current Concepts
3. Inspiration for the Future

Big Ideas

- Computer Go work: **Big Ideas** and **Essential Details**
- Natural tendency: Use big ideas of others and focus on essential details
- Essential details: Reach or somewhat surpass current state-of-art level
  Immediate reward
- Big ideas: Large leap in possibilities
  1% success rate (optimistic)

Big Ideas

- Computer Go work: **Big Ideas** and **Essential Details**
- Natural tendency: Use big ideas of others and focus on essential details
- Essential details: Reach or somewhat surpass current state-of-art level
  Immediate reward
- Big ideas: Large leap in possibilities
  1% success rate (optimistic)
- Still, please think of big ideas and work on them!
  - They are your new contributions to human understanding
  - You will be famous and your program will get very strong :-)

Information Sharing

- Common ways to improve a Computer Go program: Go knowledge, opening books, machine learning improvements, . . .
- Today's big idea: **Information sharing**

- By default, MCTS spends a lot of CPU resources on a simulation and then uses only a single number — result of the simulation!
- By default, MCTS does not share any data between branches of the tree, even though many areas of the board can be mostly independent
- Let's fix that!

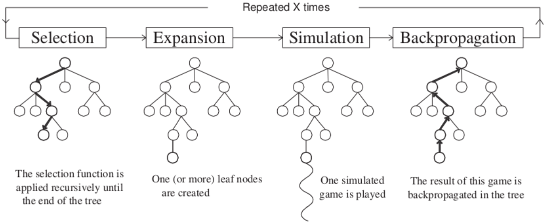
- Information flow: From simulations to tree, from tree to simulations, between tree branches, between simulations

Framework: Monte Carlo Tree Search

- Probabilistic minimax tree, each tree has expected win probability μ
- Multi-armed Bandit: exploration vs. exploitation

Framework: Monte Carlo Tree Search

- Probabilistic minimax tree, each tree has expected win probability μ
- Multi-armed Bandit: exploration vs. exploitation
- Only root at the beginning, and we repeat:
  - Descending to the leaf, sons chosen by Multi-armed Bandit
  - If we visit a leaf n times, we create son nodes
  - In the leaf, we start a Monte Carlo simulation
  - The result is propagated back through the path to the root

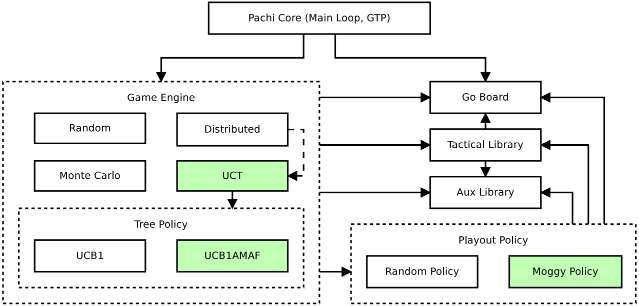


Framework: Monte Carlo Tree Search

- Probabilistic minimax tree, each tree has expected win probability μ
- Multi-armed Bandit: exploration vs. exploitation
- Only root at the beginning, and we repeat:
  - Descending to the leaf, sons chosen by Multi-armed Bandit
  - If we visit a leaf n times, we create son nodes
  - In the leaf, we start a Monte Carlo simulation
  - The result is propagated back through the path to the root
- The tree grows dynamically based on search direction
- RAVE Multi-armed Bandit: We search moves that worked well in simulations

Framework: Pachi Software

- Computer Go player
- Standard MCTS: RAVE algorithm and a set of heuristics
- Modular, fairly small; optimized C, open source



Framework: Pachi Software

- Computer Go player
- Standard MCTS: RAVE algorithm and a set of heuristics
- Modular, fairly small; optimized C, open source
- Play-testing infrastructure "autotest"
- Pattern harvesting infrastructure

Outline

RAVE

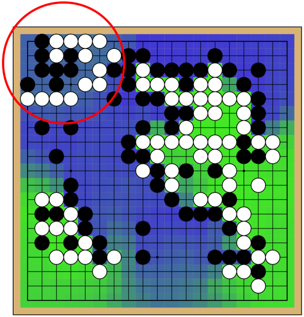
- Most strong programs are using it (except **Nomitan**!)
- Keystone of modern Computer Go, major method of information sharing; modern programs use it aggressively (even completely replacing UCT)
- Key questions of information sharing: *What information* to share, *the scope where* we share it, *how to use it?*
- **What information:** "All moves as first"
- **The scope:** Aggregating statistics by "common prefix" in the tree
- **How to use it:** The RAVE formula

$$\beta = \frac{sims_{RAVE}}{sims_{RAVE} + sims + sims_{RAVE} sims / sims_{EQUIV}}$$

$$\mu^{RAVE} = \beta \frac{wins_{RAVE}}{sims_{RAVE}} + (1 - \beta) \frac{wins}{sims} \qquad \pi_{RAVE} = \mathrm{argmax}_i \, \mu_i^{RAVE}$$

Criticality

- First presented by Rémi Coulom at UEC 2009
- Some areas are essential for winning and unsettled!
- We should "somehow detect" and "somehow use" this to focus the search better



Criticality

Point Criticality: Covariance of random variables "owning an intersection" and "winning a game"

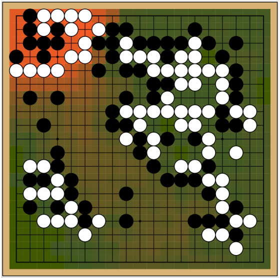
$$Cov(Owning, Winning) = \mathbb{E}Owning \cdot Winning -$$

$$-\mathbb{E}Owning \cdot \mathbb{E}Winning$$

$$C_{\mathrm{Coulom}}(x) = \mu_{win(x)} - (\mu_{b(x)}\mu_{\mathrm{b}} + \mu_{w(x)}\mu_{\mathrm{w}})$$

$$C_{\mathrm{Pell.}}(x) = \mu_{\mathrm{b},b(x)}\mu_{\mathrm{w},w(x)} - \mu_{\mathrm{w},b(x)}\mu_{\mathrm{b},w(x)}$$

$$C_{\mathrm{Pachi}}(x) = \mu_{win(x)} - (2\mu_{b(x)}\mu_{\mathrm{b}} - \mu_{b(x)} - \mu_{\mathrm{b}} + 1)$$



Criticality

- We have a *number* ($C \in [-1, 1]$)
- Scope: Global table or common prefix like RAVE
- Usage: This makes the real difference
- CrazyStone: Progressive widening criterion, pattern feature
- Orego: $\mu_{Crit} = \mu_{UCT} + 2C$ (RAVE replacement!)
- Pachi: Proportionally adding RAVE wins or losses

$$sims_{Crit} = |C_{\text{Pachi}}(x)| \cdot sims_{AMAF}$$

$$wins_{Crit} = \begin{cases} C_{\text{Pachi}}(x) > 0 & sims_{Crit} \\ C_{\text{Pachi}}(x) < 0 & 0 \end{cases}$$

$$sims_{RAVE} = sims_{AMAF} + sims_{Crit}$$

$$wins_{RAVE} = wins_{AMAF} + wins_{Crit}$$

Last Good Reply

- Let's make the simulations adaptive! (Drake et al., Henrik Baier)
- Maintain function LGR_n: $a_1, \ldots, a_n \rightarrow a^*$ that stores whether a move a^* preceded by a sequence of other moves
- Usage: If our context matches an entry in LGR_n, play the stored move! $n = 2$ is common
- Forgetting: We remove a^* if we lost the simulation
- Very nice thesis of Henrik Baier explores also (bad) performance of scoping and variations

Other

- Dynamic komi — add/substract extra virtual komi based on average result of previous simulations
- Monte Carlo point owner — simpler alternative or complement to criticality
- Some others that I have probably forgot

- The first method for information sharing you implement has by far the largest impact
- RAVE is the popular first choice, but it seems RAVE might not be so essential!

Outline

Local Trees

- Motivation: Let's combat the horizon effect!
- Idea: Aggregate *local sequences* from various branches of the Monte Carlo tree
- (L_B, L_W) pair of "local trees" storing black-first (or white-first, respectively) local sequences
- Sequences are aggregatd from all branches of the main tree
- Sequences are paused when tenuki happens in the main tree

- Ideally, if we e.g. misevaluate some corner in simulations, and keep pushing out the correct solution from the main tree, we will still learn the correct solution sequence in the local tree

Local Trees

- Big problem (as usual): How to *use* this information?
- Simple approach: Bias main tree search based on local sequences
- Experience: Difficult to overcome bias from simulations flowing in by RAVE
- How to bias the simulations based on local tree information? Not so clear!
- **Forcing:** Play out n sequences from local tree (chosen by UCT) just before a simulation begins
- Negligible improvement (for now?)

Liberty Maps

- Biasing simulations based on previous experience: Alternatives to local trees? (together with Kroon and van Niekerk)
- **Liberty maps:** A way to aggregate information on moves regarding a group *in the same situation*
- **Liberty map of group** G**:** Zobrist hash of G's liberties, taking *their* liberties into account (rotating the hash)
- $\Rightarrow$ Liberty map is a hash number (index in hash table)
- Information stored: Track success of tactical heuristics candidate moves
- Usage: Prefer moves with better success rates — either over threshold or UCB (sort of local UCT-ish transposition table arises!)
- Only small improvement (for now?)

Goal-based Search

- Generalizing liberty maps: tracking success of moves to achieve goals
- Goal (high-level): Killing or surviving with a given group
- Goal (low-level): Coloring an intersection with a given color *(what about eyes?)*
- Fill goal-achieving moves from simulation heuristics and "touching moves" anywhere in the tree
- Possible optimization: Consider only groups present in the tree as possible goals
- Opportunities for sharing results of one-time static analysis
- Usage: Heuristic choice preference, maybe also explicitly try to achieve goals in simulations?

Conclusion

- Information sharing allows us to mine maximum amount of information from simulations — best return for CPU time invested
- Our simulations could learn from past mistakes (or good choices) and eventually solve even unexpected situations
- Maybe we can approach pro strength by better ways to share information (but *essential details* are still essential, I'm sorry)
- Try out your big ideas with Pachi! ;-)

Conclusion

- Information sharing allows us to mine maximum amount of information from simulations — best return for CPU time invested
- Our simulations could learn from past mistakes (or good choices) and eventually solve even unexpected situations
- Maybe we can approach pro strength by better ways to share information (but *essential details* are still essential, I'm sorry)
- Try out your big ideas with Pachi! ;-)

Thanks!