Computing Science (CMPUT) 657
Algorithms for Combinatorial Games

Martin Müller

Department of Computing Science
University of Alberta
mmueller@ualberta.ca

Fall 2025

Temperature Discovery Search - Heuristic Search

- Motivation: solve sum games with complex subgames
- TDS so far: exact solutions for mean and temperature
  - Complete search
  - Small-enough δ
- Next: heuristic search
- Refinements, TDS+
- Experiments

Heuristic TDS - Motivation

- Exact searches do not scale well to complex subgames
- Temperatures can have large denominators, e.g. $t = k/32$ would need $\delta = 1/64$
- Temperatures can get hot, e.g. with $t = 10$, $\delta = 1/64$ the search depth becomes over $11 \times 64 = 704$
  - 11 because playing down to $t = -1$
  - Game can be even longer if we need to play out an integer game using several -1 coupons at the end

Heuristic TDS

- Approach
  - Search with larger δ
  - Guess a lower initial temperature for the stack $C(t, \delta)$
  - Limit search depth and/or time
  - Use heuristic evaluation in non-terminal leaf nodes of search
  - Find some temperature approximately in $[t(G), \hat{t}(G, p)]$,
    - This is good enough to play well: remember discussion of one-sided sente
    - Do not need to try to lower t estimate to $t(G)$
- Consequences
  - Result is approximate, not exact
  - Several re-searches may become necessary
  - Several Types of result

Heuristic Evaluation

- Exact evaluation if board is played out completely:
  - Score = difference in coupons taken
- Exact evaluation if game engine recognizes integers (e.g. territories in Amazons):
  - Score = difference in coupons taken + board evaluation (integer)
- Heuristic Evaluation in non-end position:
  - Heuristic board evaluation
  - + coupons taken
  - $\pm$ minimax value of remaining stack (depending on toPlay)
- Can also be used to speed up alphabeta: iterative deepening, move ordering
- In Amazons: use min-distance heuristic

Min-distance Heuristic

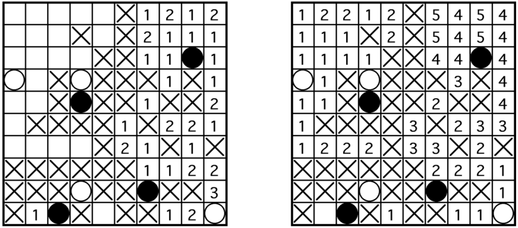


Figure 20: Min-distance function for Black and White

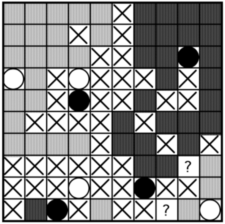
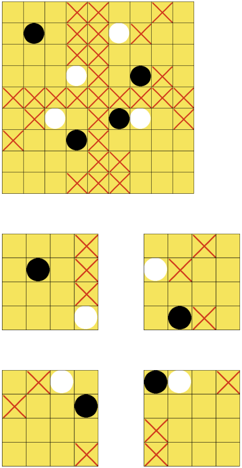


Figure 21: Evaluation using min-distance

Heuristic TDS - results and Re-search

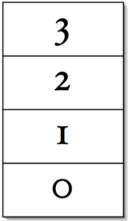
- Heuristic Search $G + C$
- Result PV $[m_1, \ldots, m_k]$
- Fail high: All m_i are coupon moves. Re-search with lower t in $C(t, \delta)$
- Fail low: m_1 is in G. Re-search with higher t in $C(t, \delta)$
- Regular: PV starts with one or more moves in C, but has at least one move in G
- Questions:
  - How to choose initial t?
  - How to choose depth, time limits?
  - How to choose δ?

Amazons example

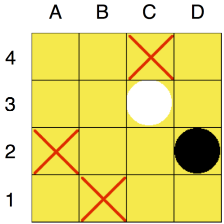


- Sum of four subgames
- Here, each subgame is 4x4, 1 queen each, 3 random obstacles
- We have a random subgame generator
- Can vary number of subgames, subgame size, obstacles, queens

Search Example

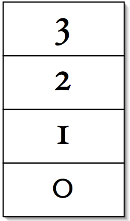


+

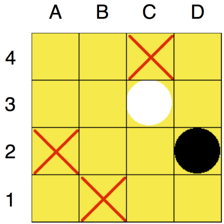


- 4×4 Amazons position G
- Coupon stack C with $t = 3$, $\delta = 1$, no negative coupons
- Minimax search of $G + C$
- Assume Black goes first

Search Example - Principal Variation

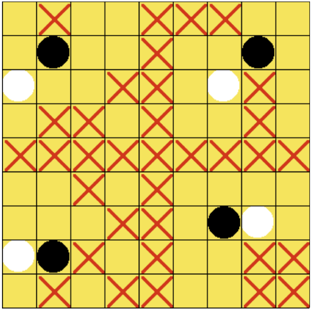


+



- *Principal variation* (PV) in alphabeta = sequence of strongest moves by both players
- $C(n)$ means: take coupon of value n
- In this example, start of PV:
  1. B: C(3)
  2. W: C3 - C2 x C3
  3. B: C(2)
  4. W: C(1)

Hotstrat-TDS Implementation Issues



- Which move to play?
- For hottest subgame: search $G + C(t, \delta)$, set t to estimated temperature, force the first move to be in G, play first move from PV
- Cache and re-use subgame temperatures (only 1 or 2 subgames change)
- Which coupon stack, which time settings to use?
- Get rough idea of t first, then refine if time
- End of game, $t = -1$: use global search to avoid zugzwangs

TDS Experiments

1. Verify exact mean and temperature by TDS
2. Measure approximation performance with larger δ
3. Evaluate depth-limited heuristic TDS in Amazons
4. Games against full-board alphabeta

Verify exact mean and temperature

- n point rooms: 1 black amazon, 1 white amazon, $n - 2$ empty
- Built complete database for $n = 4, 5, 6$, with over 4000 distinct positions
- Solved them using retrograde analysis, thermographs
- Ran TDS on all of them. t and δ set as in paper
- All results agree with DB

Measure approximation performance with larger δ

- Same test set
- Vary δ from 1, 1/2, ..., down to value needed in worst-case by theory
- Measure average and maximum errors for mean and temperature
- Very good approximations even for $\delta = 1/2$!

Average Approximation Error with larger δ

δ \ Size	4	5	6
$\delta = 1$	0.155 / 0	0.334 / 0.086	0.306 / 0.150
$\delta = 1/2$	0 / 0	0.0029 / 0.024	0.0099 / 0.020
$\delta = 1/4$	**0** / **0**	0.0014 / 0	0.0016 / 0.005
$\delta = 1/8$	–	**0** / **0**	0.0008 / 0
$\delta = 1/16$	–	–	**0** / **0**

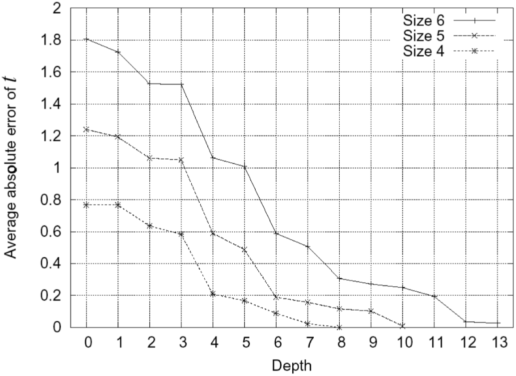
Table 1: Average absolute errors of t/μ

Maximum Approximation Error with larger δ

δ \ Size	4	5	6
$\delta = 1$	1 / 0	1.5 / 0.75	1.5 / 0.75
$\delta = 1/2$	0 / 0	0.25 / 0.25	0.375 / 0.25
$\delta = 1/4$	**0** / **0**	0.125 / 0	0.125 / 0.125
$\delta = 1/8$	–	**0** / **0**	0.0625 / 0
$\delta = 1/16$	–	–	**0** / **0**

Table 2: Maximum errors of t/μ

Evaluate depth-limited heuristic TDS in Amazons



- TDS with increasing depth limit
- Heuristic evaluation in leaf nodes
- Error decreases to 0

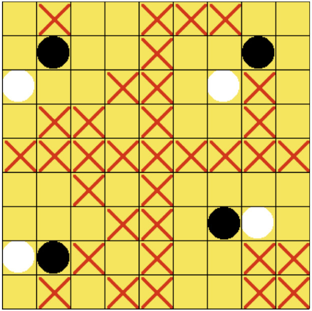
Test Games on sum games

- Players
  - Hotstrat-TDS: see next slide
  - Arrow - Amazons program doing full board alphabeta minimax search
  - For 4x4 subgames only: Hotstrat-TDS version using database with exact temperatures
- Sum game:
- Subgames size 4×4, 5×5, 6×6
- Queens and arrows in random locations
- Number of subgames: 2, 4, 6

Classic Full-Board Approaches

- Alpha-beta (Arrow program, tested)
- Monte Carlo Tree Search (Arrow2 program, not yet tested)
- Full board searches do not exploit any subgame structure
- Alphabeta scales badly with full-board branching factor, full-board search depth
- Best case $b^{d/2}$ for constant b, d
- Of course b, d not constant here, but still they are much larger than for one subgame
- I suspect Monte Carlo will behave similarly to alphabeta, maybe a bit better? **Project idea**

Sum Game Player: Hotstrat-TDS



- Hotstrat: play in subgame with highest temperature
- Hotstrat-TDS: estimate temperature of each subgame by TDS
- Example: 4 subgames
  - Top left: t=2
  - Top right: t=1
  - Bottom left: t=3
  - Bottom right: t=0
- Bottom left is hottest - play there

Results

N	4×4	5×5	6×6
2	$-1.7(2.5)$ 43%	$1.2(5.7)$ 55%	$7.8(10.2)$ 67%
4	$2.3(4.9)$ 57%	$13.5(11.5)$ 69%	$41.2(18.3)$ 90%
6	$8.1(6.2)$ 72%	$32.5(15.3)$ 88%	$81.2(26.8)$ 96%

Table 3: Game results depending on the number N and the size of the subgames. Each entry shows the mean score, the standard deviation of the score and the percentage of wins

- Advantage for Hotstrat-TDS grows quickly with both number and size of subgames
- Global search only reaches depth of 2 for larger games
- Local searches can go deeper
- Temperature-based play not perfect, but pretty good: even approximation is enough
- Advantage accumulates step by step over whole game

Summary of TDS

- Local search algorithm
- Discovers temperature by minimax search
- Exact version computes mean and temperatures
- Excellent approximation algorithm
- Hotstrat-TDS beats global alphabeta search
- Application to Amazons, future: Go

TDS+ (Zhang and Müller 2015)

- Series of five improvements to TDS
- Addresses some search inefficiencies
- Orders of magnitude improvements in speed
- Updated sum game results on much faster hardware

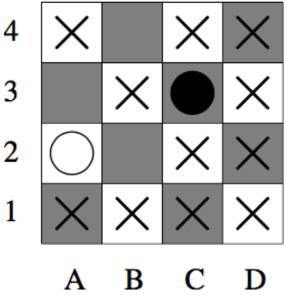
Search State

- Consider states in the search tree for $G + C$
- *Search state* in TDS:
- $S(g, c, \Delta, toPlay)$
    - g ... current game position, reached by some move sequence from G
    - c ... current remaining coupon stack
    - Δ ... aggregate score of coupons taken so far
    - *toPlay* ... color to play next

Search State and Moves

- Search state $S(g, c, \Delta, toPlay)$
- All moves change toPlay to opponent
- Move in g: changes g to g'
- New state $S(g', c, \Delta, opp)$
- Move in c: changes both c and Δ
- New state $S(g, c', \Delta', opp)$

Scaling Problems of TDS



- Very bad scaling with decreasing delta
- Bad scaling with increasing t_{max}
- Example from TDS paper
- $\mu(G) = \frac{3}{4}$, $t(G) = \frac{5}{4}$
- δ required from theory: $\delta = \frac{1}{8}$

Example - Scaling with Delta

- PV for $\delta = 1/2$, $t_{max} = 2 + \delta$
- 9 ply search, fast
- Pretty good approximation to $t(G) = \frac{5}{4}$

1. $C(\frac{5}{2})$ **2.** $C(\frac{4}{2})$
3. $C(\frac{3}{2})$ **4. A1–A2$\times$B1**
5. $C(\frac{2}{2})$ **6.** $C(\frac{1}{2})$
7. $C(0)$ **8.** $C(-\frac{1}{2})$
9. C2–B3$\times$C2.

- Exact search, $\delta = 1/8$
- Deep 23-ply search
- Long sequences of coupons
- t_{max} estimate too high: many coupons at start

1. $C(\frac{17}{8})$ **2.** $C(\frac{16}{8})$ **3.** $C(\frac{15}{8})$
4. $C(\frac{14}{8})$ **5.** $C(\frac{13}{8})$ **6.** $C(\frac{12}{8})$
7. $C(\frac{11}{8})$ **8. A1–A2$\times$B1** **9.** $C(\frac{10}{8})$
10. $C(\frac{9}{8})$ **11.** $C(\frac{8}{8})$ **12.** $C(\frac{7}{8})$
13. $C(\frac{6}{8})$ **14.** $C(\frac{5}{8})$ **15.** $C(\frac{4}{8})$
16. $C(\frac{3}{8})$ **17.** $C(\frac{2}{8})$ **18.** $C(\frac{1}{8})$
19. $C(0)$ **20.** $C(-\frac{1}{8})$ **21.** $C(-\frac{2}{8})$
22. $C(-\frac{3}{8})$ **23. C2–B3$\times$C2.**

Simple Tree Search Model and TDS

- Fixed branching factor b, depth d
- Best case time complexity of $\alpha\beta$: $\Theta(b^{\lceil d/2 \rceil})$
- Compare $\alpha\beta$ search of $G + C$ with $\alpha\beta$ search of G:
- b increases by one for the coupon move
- d increases massively:
  - By number of coupons taken before reaching terminal position in G
    - Can be about $(t_{max} + 1)/\delta$
  - Plus possibly several final coupons of value -1

Improving TDS

- Avoid long sequences of coupons
  - At the beginning of the search
  - After a temperature drop from a move in G
- Approach: fast pre-searches with larger δ
  - Quickly gain information about a game state
  - Set up t_{max} for the final, most expensive search as well as possible
- Algorithm-specific improvements
  - Better, specialized transposition table in $\alpha\beta$ search
  - Use properties of coupon stacks for finding more transpositions

Implementing Temperature Drops in Search

- Temperature drop from t to $t' < t$ corresponds to taking long sequence of coupons
- We will have several such cases recognized in TDS+
- How to implement?
- We chose simplest way:
- Unbranched "search": take only coupons, forbid all moves in G until $t = t'$

Five Enhancements of TDS+

- E_1: Fast Pre-Searches With Decreasing Values of δ
- E_2: Avoid Search at Impossible Temperatures
- E_3: Generalized Transposition Table
- E_4: Recursive TDS
- E_5: Improved Handling of “Pseudo-terminal” Positions

E_1: Fast Pre-Searches With Decreasing Values of δ

- Original TDS: $t_{max} = bound + \delta$
- Worst-case *bound* - game-specific limit on max. temperature
  - Amazons, *n* empty squares: *bound* = $n - 1$
- Temperature of most positions is much lower than worst-case bound
- Idea: search with a larger δ to quickly get a better t_{max} estimate

Fast Pre-Searches to Find Good t_{max}

- Notation: Search $TDS(G, \delta, t_{max})$
- Returns t_δ, temperature estimate from search with δ-spaced coupons
- Pre-search algorithm:
- Search $G + C$ repeatedly with decreasing $\delta = 1, \frac{1}{2}, \cdots, \frac{1}{2^n}$
- Set t_{max} using the t_δ estimate of previous search:
- $t_{max} = t_\delta + 2\delta$

2δ-Conjecture

- Empirical observation: estimated temperature returned from search with large δ never underestimates by much
- Approximate temperature computed for some δ:
- $t_\delta = TDS(G, \delta, t_{max})$
- 2δ-Conjecture: $t(G)$ is upper bounded by $t(G) \leq t_\delta + 2\delta$
- This was always true in every TDS run we did, many thousands
- But maybe our test cases were too simple?
- I have no strong intuition whether conjecture is true or not

Benefits of Pre-search

- Choose lower, position-dependent t_{max}
- Fewer coupons in the final, most expensive search
- Ideal case: PV starts with single coupon, followed by a move in G
- Reduces search depth

E_2: Avoid Search at Impossible Temperatures

- Assume $\delta = 1/8$
- At negative *t*, coupon stack would be $-1/8, -2/8, -3/8, -4/8, -5/8, -6/8, -7/8, -1, -1, ...$
- Many of these *t* cannot happen in CGT, so searching them is useless.
- The only possible negative *t* are of form $-1/2^n$, so here: $-1/8, -2/8 = -1/4, -4/8 = -1/2, -1, -1, ...$
- Similarly, if we know bound on birthday $b(G) \leq n$, we can restrict set of $t > 0$ (next slide)
- As discussed before, we simply skip moves in *G* at those impossible *t*
- Amazons position *G* with *n* empty squares: $b(G) \leq n$.
- Note: we can adjust birthday bound after each move!

Possible Temperatures and Means Theorem

- Game G born by day $n \in \mathbb{N}$
- $t(G) \in T_n$:
- $T_n = \{-\frac{1}{2^b}, 0, \frac{1}{2^b}, \frac{3}{2^b}, \ldots, a + \frac{1}{2^b} | 0 \leq a \leq n-2, 0 \leq b \leq n-1\}$
- $\mu(G) \in M_n$:
- $M_n = \{0, \pm\frac{1}{2^b}, \pm\frac{3}{2^b}, \ldots, \pm(a + \frac{1}{2^b}), \pm n | 0 \leq a \leq n-2, 0 \leq b \leq n-1\}$

E_3: Generalized Transposition Table

- original TDS: standard hash table for transpositions in $\alpha\beta$
- Hash function for coupon stacks "top down": top coupon has hashcode[0], 2nd coupon hashcode[1], etc.
- No re-use for re-search with new t_{max}
- Three improvements in TDS+
  - Better hashing for coupon stacks to allow re-use
  - Deal with *graph history interaction* issue (later)
  - Generalized entries in hash table

Better Hashing for Coupon Stacks

- Zobrist hashing (standard in game tree search)
- Table of hash codes for each point $\times$ state pair on board
- Code of board = XOR of all pointwise codes
- Zobrist hash for coupon stack:
  - Map each temperature t to a hash code $h(t)$
  - Hash code of stack c: XOR codes of all coupons $C(t)$ with $t > -1$
  - Advantage: same $G + C$ hashes to same code even when starting with different t_{max}
  - Search with different δ produces different codes
  - Can keep, re-use table between searches

Generalized Table Entries

- Full state $S(g, c, \Delta, \text{toPlay})$
- What is its minimax score?
- Δ + (search score for $g + c$ with toPlay going first)
- Same search if only Δ is different - only search once!
- Always store search results for $S(g, c, 0, \text{toPlay})$
- Just remove Δ from table entries
- Update Δ incrementally during search
- Add to table lookup score

Example

- $G + C$ where $C = C_{-1}(4, 1)$
- Line 1: 1. C(4), 2. C(3), 3. play in G
- Assume this line has finished search
- The entry in table is for $S(G, C_{-1}(2, 1), \text{Black})$
- The value $\Delta = 4 - 3 = 1$ is added in search node

- Now search $G + C$ where $C = C_{-1}(2, 1)$
- We can lookup its value directly from table, no search!
- If this is some state in middle of other search, we add the new Δ from that state

E_4: Recursive TDS

- E_1 was very good to lower t_{max} at the beginning of the search
- Can do the same after each move from G to G' (some option in G^L or G^R)
- Run a pre-search with large δ to get an idea of $t(G')$
- If temperature drop: skip search by forcing unbranched sequence of coupon moves
- This can skip many coupons in the main search
- Greatly reduce search depth
- In running example: PV had 14 coupons between first and second move in G
- We can skip most of them cheaply by pre-search

Summary so far

- Heuristic TDS - excellent approximation for $\delta = 1/2$ or smaller
- Only forward search method to compute means and temperatures that works for general games (including undetected numbers, zugzwangs,...)
- Clobbers global search in sums with many hot subgames
- TDS+ addresses some search inefficiencies in original TDS
- Main problem: avoid long sequences of coupons in search
- Discussed 4 enhancements
- Still to do: GHI, fifth enhancement, experimental results

E_5: Improved Handling of "Pseudo-terminal" Positions

- Graph history interaction (GHI) problem
- How can it happen in a loop-free game? (Surprise!)
- Remark on general solution
- Simple fix for GHI in TDS+

Graph History Interaction (GHI) Problem

- Problem: same position, different value
- Usually caused by positon repetition rules, e.g. in checkers, chess, Go
- Ko in Go: same position, is capture on a legal?
- Answer depends on history
- Was same position on path leading to state or not?

GHI in TDS

- Assume game G has no history dependency (e.g. Amazons)
- GHI can appear when searching $G + C$ (!)

Pseudo-terminal Position (PTP)

- In search, how does play of $G + C$ end?
- *Normal terminal position*
  - Value of G can be statically recognized
  - Example: game over, value 0
  - Example: game-specific recognition of integers
- *Pseudo-terminal position* (PTP)
  - Both players took a -1 coupon as their last move
  - Example: Zugzwang
  - PTP evaluated as 0 by *simplicity rule* of CGT

How does PTP cause GHI?

- End of game defined by successive -1 coupons
- We have path dependence!
- Example: Coupon stack $c_{-1} = C_{-1}(-1, \delta)$, only -1 coupons left
- play $G + c_{-1}$
- Assume there are moves Black a and White b, such that order of playing them does not matter
  - E.g. both fill their territory in Amazons
- Now we can get two sequences with 1. equal game position, 2. equal stack, but 3. different evaluation!

GHI Example

Play $G + c_{-1}$, Black goes first

Line 1: ends in PTP	Line 2: does not end in PTP
1. Black a	**1.** Black $C(-1)$
2. White b	**2.** White b
3. Black $C(-1)$	**3.** Black a
4. White $C(-1)$	**4.** White $C(-1)$

- Line 1: evaluated as 0 by *simplicity rule*
- Line 2: Play will continue
    - Might have any minimax score, not necessarily 0
    - Example: Amazons, other moves (e.g. on integers) may still exist for one or both players

Discussion

- Both sequences result in identical board positions G'
- Both sequences result in identical stack c_{-1}
- There *is* a hidden loop here:
- A move from c_{-1} leads back to c_{-1}, since we have arbitrary many -1 coupons

Why was this Not a Problem in Original TDS?

- In TDS, transposition table only used very conservatively
- States reached after consecutive -1 coupons never stored or looked up in table
- In contrast, TDS+ uses tables in every search step, stores everything

General GHI Solution

- Kishimoto (PhD 2004) and me developed first efficient GHI solution
- Applied in Life and Death solver, and in proof that checkers is a draw (Schaeffer et al)
- Several algorithmic ideas to solve such games about as efficient as when no GHI present
- Not needed here, a much simpler fix suffices

GHI Fix for TDS+

- The only history dependence is from most recent moves being -1 coupons
- Just extend the state and store how many were taken
- State without handling GHI: $S(g, c, \text{toPlay})$
- State with handling GHI: $S(g, c, \text{toPlay}, n)$
  - Where $n \in \{0, 1, 2\}$ is number of latest moves which were -1 coupons
  - Keep counter for n during search
  - Any non-coupon move resets the counter
  - States with $n = 2$ are terminal with value 0

GHI Example Revisited

- Play $G + c_{-1}$, Black goes first
- **1.** Black a **2.** White b **3.** Black $C(-1)$ **4.** White $C(-1)$
  - Resulting state $S(G', c_{-1}, \Delta, \mathit{Black}, \mathbf{2})$
- **1.** Black $C(-1)$ **2.** White b **3.** Black a **4.** White $C(-1)$
  - Resulting state $S(G', c_{-1}, \Delta, \mathit{Black}, \mathbf{1})$
- Recognized as different
- Search is not stopped after line 2 because it no longer confuses it with PTP terminal state after line 1

Experiments

- Evaluate all 5 enhancements
- Define a standard test set
- Measure individual, and subsets of enhancements
- Measure scaling with time limit
- Measure scaling with smaller δ
- Measure approximation error as function of time limit
- Re-run sum game experiments

Enhancements and their Dependencies

- E_1: Fast Pre-Searches With Decreasing Values of δ
- E_2: Avoid Search at Impossible Temperatures
- E_3: Generalized Transposition Table
- E_4: Recursive TDS: requires both E_1 and E_3
- E_5: Improved Handling of PTP States: requires E_3

Standard Test Set

- TDS paper: complete database with 4, 5, 6 squares (2, 3, 4 empty)
- This paper: subset sampled from complete database of 4×4 Amazons positions with one queen each
- 600 positions total, randomly sampled from each "layer":
  - 17 cases with two empty squares
  - 50 test cases each for 3 to 13 empty squares
  - 33 cases with 14 empty squares
- We know exact means, temperatures from database

Experiment: Coverage for selected subsets of TDS enhancements

Time	1s	2.5s	10s	25s	100s	250s
TDS	69	73	73	74	76	76
TDS_1	78	88	100	107	116	116
TDS_2	69	73	73	75	76	76
TDS_3	73	76	85	95	117	117
TDS_{134}	**81**	**108**	**129**	**135**	**137**	**141**
TDS_{35}	73	76	86	96	117	117
TDS_{13}	99	117	130	131	136	141
TDS_{235}	73	77	91	102	117	117
TDS_{1345}	82	109	129	135	137	141
TDS_{12}	78	89	100	108	116	116
TDS_{1235}	**96**	**118**	130	135	150	157
TDS_{1234}	83	111	**133**	**136**	**153**	157
TDS_{12345}	82	111	**133**	**136**	**153**	**159**

Table: Coverage (number solved) as function of time limit.

Discussion

- Experiment with exact search
- Old TDS scales poorly with time limit
- TDS_1 solves many more cases already
- TDS_1 scaling at higher time limits is not good
- TDS_2: E_2 alone helps little
- TDS_{1345} vs TDS_{12345}: E_2 helps a lot for more complex test cases, at higher time limits

Discussion (2)

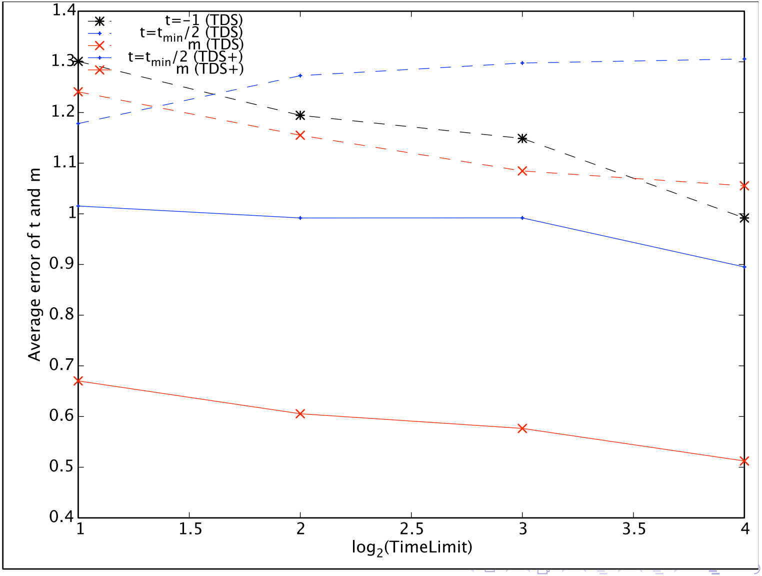
- E_3 by itself is similar to E_1
- Complementary strengths: see TDS_{13} vs TDS_1, TDS_3
- Adding E_4: strong improvement over TDS_3 but not over TDS_{13}
- TDS_{12345} better vs TDS_{1235} for high temperature test cases
- E_5: improvement, but below 1% in runtime

Experiment: approximate TDS vs approximate TDS+

$\delta = 1$	1s	3s	10s	30s	100s
TDS	85	165	188	214	246
TDS+	**178**	**195**	**240**	**261**	**305**
$\delta = 1/2$	1s	3s	10s	30s	100s
TDS	84	143	165	180	197
TDS+	**156**	**171**	**190**	**232**	**257**
$\delta = 1/4$	1s	3s	10s	30s	100s
TDS	79	95	131	156	170
TDS+	**116**	**155**	**175**	**195**	**239**
$\delta = 1/8$	1s	3s	10s	30s	100s
TDS	45	78	85	102	130
TDS+	**102**	**146**	**164**	**180**	**194**

Table: Coverage for approximate TDS and TDS+.

Experiment: approximation errors for temperature and mean as function of time limit



Discussion

- TDS+ approximates both mean and temperature better than TDS
- How to approximate t what search times out?
- Good case: if PV contains move in G: use coupon value just before
- Only coupons in PV: tried different heuristics, $t = t_{min}/2$ is good

Experiment: Sum Games

- Players:
  - Arrow, full-board $\alpha\beta$
  - Hotstrat-TDS
  - Hotstrat-TDS+
- 10 seconds per move

Hotstrat-TDS+ vs Arrow

N	4×4	5×5	6×6
2	-2.20(6.06) 44.0%	-1.62(8.95) 52.3%	0.58(11.8) 57.3%
4	-2.60(8.49) 49.3%	2.54(12.3) 58.6%	25.4(19.7) 77.5%
6	-1.58(10.1) 50.1%	16.4(16.9) 72.8%	53.9(25.4) 86.8%

Table: Results depending on number N and size of subgames.

- All numbers from Hotstrat-TDS+' point of view:
- mean score
- (standard deviation of score)
- win percentage

Discussion

- Hotstrat-TDS+ improves strongly with size and number of subgames
- For 4×4 subgames, full board $\alpha\beta$ is slightly superior
- Compared with 2004 TDS experiment: $\alpha\beta$ much improved for simple subgames due to extra search depth reached
- $\alpha\beta$ plays perfectly in the limit, Hotstrat does not
- Superior scaling of local search remains very clear for more complex subgames

Hotstrat-TDS+ vs Hotstrat-TDS

N	4×4	5×5	6×6
2	9.77(5.53) 82.5%	22.2(9.26) 88.2%	43.2(13.5) 91.4%
4	19.7(7.59) 90.0%	39.7(12.2) 94.7%	61.1(16.2) 97.7%
6	29.9(9.85) 93.3%	50.7(15.5) 96.7%	76.6(21.6) 98.8%

Table: Results Hotstrat-TDS+ vs Hotstrat-TDS.

- Hotstrat-TDS+ much better than Hotstrat-TDS
- TDS+ computes better approximations in the same time
- Advantage increases with both size and number of subgames

Future Work

- Prove or disprove 2δ-Conjecture
- Use TDS+ to generalize Kao's Mean and Temperature Search (MTS)
- extend TDS+ to Go endgames, deal with *ko*
- **hybrid algorithm**, combine local TDS+ with shallow global search as in (Müller + Li) paper

Summary

- 5 enhancements greatly speed up TDS
- E_4 promises to scale well for even larger subgames
- Still lots of room for improvement
  - Only of 159 of 600 test cases solved exactly within 250 seconds
  - Only 305 of 600 solved approximately within 100 seconds, even with large $\delta = 1$