Temperature Discovery Search

Martin Müller and **Markus Enzenberger** and **Jonathan Schaeffer**
Department of Computing Science, University of Alberta
Edmonton, Canada T6G 2E8
{mmueller,emarkus,jonathan}@cs.ualberta.ca

Abstract

Temperature Discovery Search (TDS) is a new minimax-based game tree search method designed to compute or approximate the *temperature* of a combinatorial game. TDS is based on the concept of an *enriched environment*, where a combinatorial game G is embedded in an environment consisting of a large set of simple games of decreasing temperature. Optimal play starts in the environment, but eventually must switch to G. TDS finds the temperature of G by determining when this switch must happen. Both exact and heuristic versions of TDS are described and evaluated experimentally. In experiments with sum games in *Amazons*, TDS outperforms an $\alpha\beta$ searcher.

Keywords: Temperature Discovery Search, combinatorial games, $\alpha\beta$ algorithm, Go, Amazons

Introduction

The effort required to solve games by minimax search typically increases exponentially with the required search depth, even when all standard search enhancements are used (Schaeffer 1989). Improvements are possible for games that exhibit additional structure. One example are *combinatorial games* (Berlekamp, Conway, & Guy 1982), which can be broken down into a *sum* of independent *subgames*.

Go endgames can be analyzed as combinatorial games. They often consist of several independent fights to refine the boundaries of loosely surrounded territories. As another example, Figure 1 shows a position G from the game of *Amazons* that decomposes into a sum of four subgames $G_1 + G_2 + G_3 + G_4$. In Amazons, playing pieces called amazons move like chess queens and try to block the other player by shooting arrows that block empty squares. Areas that become completely blocked from each other become independent subgames. In the figure, each of the four corners forms one subgame. Each corner is separated from the rest of the board by a solid wall of blocked squares.

One important goal of algorithms for combinatorial games is to replace global minimax search by local analysis of single subgames as much as possible. Because of the exponentially growing cost of search, and because the length of local move sequences is typically much shorter than playing out the whole sum game, even methods that perform substantial analysis of each local game can still be much faster than global search. For example, in Figure 1 there are several hundred legal moves and 39 empty squares, so the game can last up to 39 moves. Local analysis reduces both the width and the depth of the search by a factor of 4 or more. This is a massive reduction in the size of the search space.

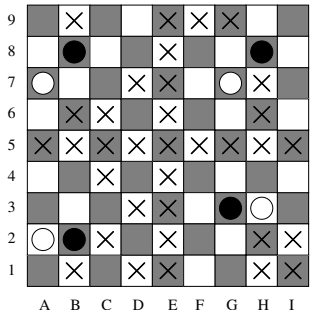

Figure 1: An *Amazons* position with four subgames of size 4×4 (Black to move)

An exact local search method that was applied to late-stage Go endgames is *decomposition search* (Müller 1999). Approximation algorithms must be used when the exact solution of local problems or the global combination of local solutions is too costly. Methods based on the concepts of *mean* and *temperature* (Berlekamp, Conway, & Guy 1982; Berlekamp 1996) can yield very good play, with bounded error. The mean of a subgame measures how advantageous for a player this game is "on average", when played as part of a sum of games. The temperature measures the urgency of playing first in a particular subgame. These two numbers are usually computed "bottom-up" by retrograde analysis, which requires building the complete local game graph.

This paper introduces *Temperature Discovery Search (TDS)*, a method for discovering the temperature of a local game by a forward minimax search. The method is more general and more practical than Kao's *Mean and Temperature Search* (Kao 2000). Kao's method applies to the restricted class of binary game trees, which allow only a single move for each player in a local game position. TDS has

no such restriction. In many combinatorial games, including Go endgames and Amazons, there are local positions with many plausible moves for both players. Kao's method also requires a search to the end of the game, whereas TDS can be used as an approximation method with a given depth or time limit.

The contributions of this paper are:

- A novel use of AI search techniques to solve a problem in the domain of combinatorial games, and the novel use of combinatorial game theory to enhance the performance of an AI game-playing program.
- The first general *forward* search method to compute the mean and temperature of any finite loop-free combinatorial game.
- The first known systematic method to *approximate* mean and temperature by forward search.
- Several techniques to control the trade-off between speed and accuracy, and their empirical evaluation.
- A series of experiments in practical play of hot sum games, showing that TDS in combination with the *Hotstrat* algorithm convincingly outperforms an $\alpha\beta$ searcher.

Combinatorial Games

A major difference between normal $\alpha\beta$ minimax search and local analysis of a sum game is that while $\alpha\beta$ solves a single optimization problem, in a sum game there is a trade-off between a local gain and the right to play first in another game. Combinatorial game theory analyzes subgames under the assumption that there are (many) other subgames around, and seeks to optimize the score in the sum game, not just in the current subgame. The gain from playing in one subgame must be compared against the potential loss from letting the opponent play first in the remaining subgames. Combinatorial game theory provides tools to quantify this trade-off. In a real-world sum game, the other subgames can be arbitrarily complex, and therefore evaluating the trade-offs precisely is hard. In contrast, a locally restricted $\alpha\beta$ search would optimize a player's local profit but ignore the trade-off. This leads to poor play in many situations.

A middle ground is represented by the concept of a standardized *enriched environment* (Berlekamp 1996), in which each subgame is analyzed. Traditionally, players are called Left, who prefers positive values, and Right, who prefers negative values. An enriched environment consists of many *elementary switches*, simple subgames of the form $v|-v$, with a large number of values v. In $v|-v$, Left to move can gain v, while Right to move can gain $-v$. The mean is $\mu(v|-v) = 0$ and the temperature is $t(v|-v) = v$ (Berlekamp, Conway, & Guy 1982). Sum games consisting only of switches are easy to play. Playing a switch with highest temperature v is always an optimal move.

In a sum game consisting of a single, potentially complex, subgame G plus an enriched environment, the trade-off now becomes much simpler to analyze: the choice is between playing in G and playing the switch with highest value v in the environment. If v is very large, each player will take this profit rather than play in G. As the available values v become smaller, eventually a player will prefer to move in G instead.

For example, consider playing a game and having a choice of making a move (and thereby improving your position) or taking some money. How valuable is it to Left to have the right to move? Assume the money is a stack of coupons, of decreasing value \$10, \$9,... and the advantage Left gets by playing on the board is worth \$5 to Left. In this case Left would take the coupon, making the highest remaining coupon valued at \$9. Right would do a similar analysis, and conclude the position is (say) worth \$8, so Right takes the \$9 coupon leaving \$8 as the top coupon value. Nothing has changed on the board, so Left takes the \$8 coupon, leaving a \$7 top coupon. Now Right makes a move since moving is worth more (\$8) that taking the next coupon (\$7). If you view the coupon stack as being the value of moving in a different subgame, then the top value in the stack will decrease to the critical point where it becomes more profitable to move in the current game than to move in a different subgame. This critical point is a representation of how urgent it is to play in that subgame; this procedure finds a temperature where playing in the game is optimal.

Let G be a finite loop-free combinatorial game and p be a player. In the view of thermography (Berlekamp, Conway, & Guy 1982), a player should play in G at a temperature $t(G)$. However, in so-called *sente* situations a player has a threat that makes it possible to play in G at a higher temperature. Let $\hat{t}(G,p)$ be the highest temperature such that player p can play in G without a loss in the thermographic sense. For loop-free games, this means p can play in G at any temperature in the range $[t(G)\ ..\ \hat{t}(G,p)]$. For example, the game $G = 10|0||0$ has a temperature $t(G) = 0$, but $\hat{t}(G, \textit{Left}) = 5$. Left's move to $10|0$ is a threat that *raises the temperature*. Left can play at any temperature t in the range $[0, 5]$. However, $\hat{t}(G, \textit{Right}) = 0$, so Right should not play at a temperature above 0 here.

In the first phase of TDS, a single search is used to identify a first temperature estimate t, which is guaranteed to lie within the interval $[t(G)\ ..\ \hat{t}(G,p)]$. In a second phase, further searches are performed that successively lower the estimate t until $t = t(G)$.

Coupon Stacks

The concept of a *coupon stack* was introduced by Berlekamp in the context of analyzing Go endgames (Spight 2002). It is used to implement an enriched environment.

Definition: A *simple coupon stack* $C(t_{max}, \delta)$, where $t_{max} = n\delta$ for some integer $n \geq 0$, is the sum of elementary switches of temperatures $\delta, 2\delta, \cdots, t_{max}$. Each such switch is called a *coupon*. In $C(t_{max}, \delta)$, $t_{max} > 0$, either player can take the coupon of value t_{max}, which changes the score of the game by t_{max} in the player's favor. The move leads to a smaller stack $C(t_{max} - \delta, \delta)$. A stack $C(0, \delta) = 0$ is said to be empty and neither player has a move.

A simple coupon stack $C = C(t_{max}, \delta)$ has temperature $t(C(t_{max}, \delta)) = t_{max}$ and mean $\mu(C(t_{max}, \delta)) = 0$. The *left score*, the minimax score with Left going first, is $V(C, \textit{Left}) = \lceil \frac{n}{2} \rceil \delta$ and the *right score*, the minimax score with Right going first, is $V(C, \textit{Right}) = -\lceil \frac{n}{2} \rceil \delta$.

In a sum $G + C$, an optimal player will play in G only if $t(C) \leq \hat{t}(G, p)$. In the range $t(G) \leq t(C) \leq \hat{t}(G, p)$, playing in C and playing in G are both optimal.

Games played on a 19×19 Go board plus a coupon stack $C(20, \frac{1}{2})$ have been used in "Environmental Go" to gain insight into how professional players estimate the temperature of Go endgames (Spight 2002).

Extended Coupon Stacks for Negative Temperatures

For games where all positions of temperature 0 or below can be evaluated statically, a simple coupon stack $C(t_{max}, \delta)$ is sufficient. However, in general play has to continue at negative temperatures down to $t = -1$. This situation can happen in games with difficult to evaluate endgame positions, such as Go or Amazons. One specific example are so-called *zugzwang* positions such as $-4|3 = 0$ where neither player wants to move.

Definition: An *extended coupon stack* $C_{-1}(t_{max}, \delta)$ can be constructed from a simple coupon stack $C(t_{max}, \delta)$ by adding coupons of value $0, -\delta, -2\delta, \cdots, -1$, followed by a sufficiently large number k of further coupons of value -1, and a final coupon of value $-\frac{1}{2}$ as the bottom-most coupon. For any $\delta = \frac{1}{2m}$, with integer $m > 0$, this construction avoids bias and ensures that $V(C, p) = V(C_{-1}, p)$ for both players p. The number k of extra coupons should be chosen in a game-dependent way, large enough such that the final $-\frac{1}{2}$ coupon is never taken and the stack never becomes empty.

Temperature Discovery Search

Temperature Discovery Search (TDS) is a standard $\alpha\beta$ minimax search of the sum of a game G and a (simple or extended) coupon stack C. The important issue is how to use the information returned by the search to determine the mean and temperature of G.

Searching a Sum Game $G + C$

The $\alpha\beta$ search process for TDS in a sum game $G + C$ proceeds as follows:

Move generation Possible moves in $G + C$ are all moves in G plus one move in C, taking the top coupon (unless the stack is an empty simple coupon stack).

Terminal Positions Two types of positions are leaf nodes in the search: in a *normal terminal position* the value of G can be statically recognized. In a *pseudo-terminal* position both players have taken a -1 coupon as their last move, indicating their unwillingness to continue play. Examples of such positions are *zugzwangs* such as $-4|3$, where both players would lose points by playing on.

Evaluation The evaluation of a position $G+C$, with player p to play, is the sum of three components: evaluation of G, coupons taken, and remaining coupons in C.

1. Evaluation of G: Normal terminal positions are evaluated by their exact integer value. Nonterminal positions in G can be evaluated by a heuristic evaluation function $h(G)$ to improve move ordering or enable heuristic TDS (see below). Pseudo-terminal positions are not evaluated heuristically by the playing program. Following the *simplicity rule* of combinatorial game theory (Berlekamp, Conway, & Guy 1982), such positions are assigned a value of 0.
2. Coupons taken: The evaluation is the sum of all coupons taken by Left minus the sum of all coupons taken by Right.
3. Remaining coupons: The evaluation is $V(C, p)$. This assumes that both players take the remaining coupons in turn, beginning with the player to move.

Example

Figure 2 shows a local Amazons position G with Black = Left to move first. The search of $G + C$ with an extended coupon stack $C = C_{-1}(t_{max} = \frac{17}{8}, \delta = \frac{1}{8})$ yields the minimax score $V(G + C, \textit{Left}) = \frac{15}{8}$. This particular choice of t_{max} and δ will be motivated in the description of Experiment 1 below. Taking a coupon of value v is represented by $C(v)$ and an Amazons move by specifying the squares as **from** $-$ **to** $\times$ **arrow**. The principal variation (best line of play–PV) for Figure 2 is:

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. A2–A3×B2 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. C3–B4×C3**.

The value $\frac{15}{8}$ of the leaf position in this line is the sum of the final position value (White has 1 square; $-\frac{8}{8}$ from Black's point of view), the difference in coupon values (Black has $\frac{21}{8}$ more), and the value of the remaining coupons ($\frac{2}{8}$ in Black's favor). Most moves are coupon moves that lower the value of the coupon stack until at move 8 and again at move 23 it becomes advantageous for a player to make a move.

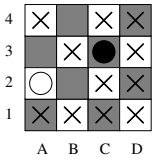

Figure 2: Example Amazons position (Black to move)

Determining the mean: Assume that p is the player to move. According to a theorem in (Berlekamp 1996), if the value δ in C is sufficiently small, then $V(G + C, p) = \mu(G) + V(C, p)$. Since $V(C, p)$ is known (see above), $\mu(G)$ can be determined. In the example, $V(C, \text{Left}) = \lceil \frac{17}{2} \rceil \cdot \frac{1}{8} = \frac{9}{8}$ (17 coupons split two ways, each of value $\frac{1}{8}$), $V(G + C, \text{Left}) = \frac{15}{8}$ and therefore $\mu(G) = \frac{15}{8} - \frac{9}{8} = \frac{3}{4}$.

Determining the temperature: Observe the move sequence in the principal variation returned by the search. If $t(C) > \hat{t}(G, p)$, then the first moves in the PV will all be

in C. At some time when $t(G) \leq t(C) \leq \hat{t}(G,p)$, the first move in G will appear in the PV. This can be used as a starting point of an iterated search process to determine $t(G)$. $t(G)$ is the lowest temperature at which optimal play can switch from C to G. In the example, the first move in G was played between the coupons of value $t = \frac{11}{8}$ and $t - \delta = \frac{10}{8}$. The first estimate of $t(G)$ is set to t, since $t(G) \leq t$. If the minimax score stays the same even if the next lower coupon of value $t - \delta$ is played before the first move in G, then the temperature estimate can be lowered by δ. To check this, TDS is modified to play all the coupons down to $C(t - \delta)$ before moves in G is allowed. If the minimax score remains the same, then taking $C(t-\delta)$ first was not a loss, compared the previous search where G was played before $C(t - \delta)$. Therefore the estimate can be decreased to $t := t - \delta$, and the process is repeated. Otherwise, if the minimax score changes, then taking $C(t-\delta)$ was nonoptimal, and the temperature is $t(G) = t$.

In the example, the first search returns $t = \frac{11}{8}$ by scanning the PV given above. In the second search, all coupons down to $C(t - \delta) = C(\frac{10}{8})$ are taken first. The minimax value of this search is also $\frac{15}{8}$, so $C(\frac{10}{8})$ was also an optimal play and the estimate is lowered to $t := t - \delta = \frac{10}{8}$. In the third search, coupons down to $C(\frac{9}{8})$ are taken first. The minimax value of this search is $\frac{13}{8}$, less than before. Taking the coupon $C(\frac{9}{8})$ before playing in G was a loss. The temperature is $t(G) = t = \frac{10}{8}$.

Heuristic TDS

Time- or depth-limited searches with TDS can be used to approximate the temperature of a game. As in any heuristic $\alpha\beta$ search, this introduces evaluation errors. Depending on the coupon stack used and the PV returned from the search, different outcomes are possible. Several searches with different coupon stacks may be required to find an approximate temperature.

Assume that TDS of a sum $G + C$ results in a PV consisting of k moves, $[m_1, \ldots, m_k]$. If the PV contains at least one move in G, determine i such that m_i is the first move in G. Otherwise, if all PV moves are in C, set $i = k + 1$. The outcome of a single search can be classified as follows:

fail high If either $i = k+1$ (all moves in the PV are coupon moves), or the last coupon of C was played before any move in G.

fail low m_1 was played in G. The temperature of G may be higher than $t(C)$.

regular Neither fail high nor fail low, $1 < i \leq k$. The PV starts with one or more moves in C followed by a move in G.

The fail high and fail low cases require additional search(es) with a different stack to find the move to play.

Hotstrat-TDS: a TDS-based Heuristic Sum Game Player

A simple way of using approximate temperatures in a playing program is to adapt the combinatorial game strategy of *Hotstrat* (Berlekamp, Conway, & Guy 1982), replacing the unknown exact temperatures by estimated temperatures computed by TDS. Given a sum game $G = G_1 + \ldots + G_n$, the temperature $t(G_i)$ of each subgame is estimated by TDS and a subgame G_{hot} with highest estimated temperature is chosen.

A best move in G_{hot} at temperature $t = t(G_{hot})$ is found by re-running TDS on G_{hot} with a stack $C_{-1}(t, \delta)$. Move generation is modified to force the first move to be in G_{hot}, not in C.

As an optimization, since each move changes only one subgame, the temperature of the other unchanged subgames can be cached and reused for the upcoming moves. To try to avoid playing into zugzwang positions at the end of the game, when the temperature estimate of all subgames is -1, a simple global minimax search is performed.

As an example of how Hotstrat-TDS works, consider playing the four subgames in Figure 1. with Black to play. TDS approximates the temperatures of the four subgames as $t_{\text{top left}} = 2$, $t_{\text{top right}} = 1$, $t_{\text{bottom left}} = 3$, $t_{\text{bottom right}} = 0$. Therefore Hotstrat selects the bottom left as the hottest game, and TDS finds A2–B3×A3 as a best move on the board at temperature 3.

Experiments

In the experiments, TDS is tested in the game of Amazons (Lieberum 2003). Amazons endgames are an ideal environment for evaluating this search method. Unlike Go, Amazons is a pure combinatorial game without the complications caused by *ko* (local position repetitions). Amazons is an attractive and difficult game on its own, with board positions that naturally split up into sum games in the course of a game. Furthermore, for small positions exact mean and temperature data is available in the form of databases. Using retrograde analysis, Theodore Tegos computed all Amazons positions on a subset of a 4×4 grid containing one amazon of each player (Tegos 2002).

All experiments were performed on a 2 GHz Athlon XP.

Experiment 1: Comparing TDS with Exact Mean and Temperature Data

The following properties of Amazons were used to set up the first and second experiment: A n point room containing 2 amazons has $n - 2$ empty points, limiting the maximum game length to $n - 2$ moves. In such a game, all means and temperatures must be integral multiples of 2^{3-n}. The maximum possible temperature of these games is $n - 3$, so $t_{max} = n - 3 + \delta$ was used. According to (Berlekamp 1996), choosing $\delta = \frac{1}{2} \times 2^{3-n} = 2^{2-n}$ is sufficient to determine means and temperatures precisely.

The first experiment compares the mean and temperature computed by TDS against the exact database values from (Tegos 2002). For all 132 rooms with size 4, all 690 rooms of size 5 and all 3330 rooms of size 6, the TDS results for both mean and temperature agreed with the database values.

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

Table 1: Average absolute errors of t/μ

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

Table 2: Maximum errors of t/μ

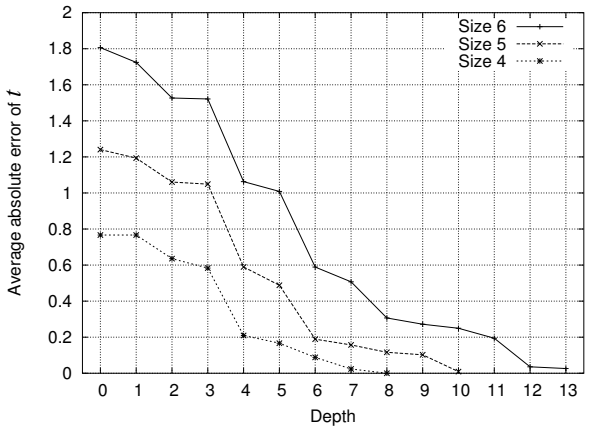

Figure 3: Average absolute errors of t for depth-limited search for subgame sizes 4 to 6

Experiment 2: Measuring the Effect of Increasing δ

Tables 1 and 2 show the effects of using values of δ larger than 2^{2-n}. Increasing δ introduces errors, but reduces the search depth and speeds up the search because far fewer coupons must be played. The experiments used all the rooms of size 4, 5 and 6 from the database. For each test position, two searches were performed, one for each player moving first. The mean and temperature estimates were set to the average of the two results, which greatly improved the accuracy. A table entry contains the error in the temperature estimate followed by the mean estimate. Table 1 shows the average absolute error and Table 2 the maximum error over the test sets.

The tables show a clear correlation between the size of δ and the accuracy of the computed means and temperatures. However, even for $\delta = \frac{1}{2}$ the average error is quite small.

Experiment 3: Depth-limited Search

An alternate way of reducing the cost of a TDS search at the expense of some accuracy is to limit the search depth. In contrast to the previous experiments, the errors in the heuristic evaluation function $h(G)$ influence the result of these searches. Figure 3 shows the average absolute error of t for fixed $\delta = \frac{1}{2}$ and varying search depth. As expected the average error decreases with deeper searches for two reasons. First, larger subgames need deeper searches because more board moves can be played in the subgame. Second, the maximum theoretically possible temperature grows with the size of the subgame and therefore a coupon stack with a higher t_{max} has to be used.

Experiment 4: Playing Sum Games against a Minimax Player

This experiment pits the Hotstrat-TDS player described above against a normal global $\alpha\beta$ search program in a sum of Amazons positions. Both programs share the same core $\alpha\beta$ search engine with standard performance enhancements and the same game-specific code such as move generation and evaluation function. The program used, *Arrow*, took third place at the Second Open Computer-Amazons Championship in 1999. Compared to the current top programs, it is weak in the opening and middle game, but competitive in the endgame phase due to its special endgame knowledge. Sums of small subgames have characteristics similar to endgame positions.

In the experiments, the number of subgames in a sum game varies from 2 to 6 in increments of 2, and the size of subgames from 4×4 to 6×6. Two amazons of different color and three arrows were placed on randomly chosen squares in each subgame. Figure 1 shows such a sum game, with four subgames of size 4×4. Usually, Amazons is played on a single 10×10 board with 4 amazons each.

For each combination of number and size of subgames, 200 random positions were created and played twice such that each player played each color once. Games were played under tournament-like conditions, but with a shorter time limit of 10 seconds per move. The score of a game is defined as the number of moves that the winning player can still make on the board when the opponent runs out of moves.

Table 3 shows the results. The average score of Hotstrat-TDS increases quickly with the number and the size of the subgames. In positions with two 4×4 subgames Hotstrat-TDS performs slightly worse than the global $\alpha\beta$ player. In this case the global search can search deeply enough to reach close to terminal positions, whereas the overhead of multiple searches for determining the temperature and best move with Hotstrat-TDS are a disadvantage here. In all other scenarios in this experiment, Hotstrat-TDS outperforms the global $\alpha\beta$ player. The advantage increases greatly as there are more and larger subgames.

Because of the reduced search width, local search in TDS can go deeper than global search. Even the simple approximation of the value of playing in a subgame by a single number, the estimated temperature, suffices to outplay global $\alpha\beta$ convincingly under tournament conditions. An alternative is to do local $\alpha\beta$ searches, to compute two local minimax values, one for each player going first, and then play a move in

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

the subgame with the largest difference in minimax values. This is an inferior strategy, both experimentally and theoretically, for the reason outlined earlier, that it maximizes local profit but ignores global trade-offs.

Figure 4 illustrates some differences between Hotstrat-TDS and global $\alpha\beta$. In this position, there are several promising places to play. However, with a branching factor of several hundred and a 10 second time limit, $\alpha\beta$ search reaches only a search depth of 4 ply, and has to rely on a heuristic evaluation for many volatile positions. In contrast, TDS can search up to 9 ply locally. Even if many of these moves are coupon moves, the evaluations are closer to terminal positions, and they are all for the same local area, so they are more reliable for both reasons.

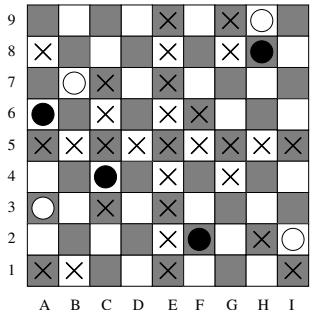

Figure 4: Sample game situation (Black to move)

Conclusions and Future Work

TDS is the first *forward* search algorithm that can compute the mean and the temperature of a combinatorial game, as opposed to the usual bottom-up analysis. A major practical advantage of this method is that it can be used to approximate these values with high accuracy by resource-limited searches. TDS is shown to be successful both in its exact version and in practical play, where positions are too complex for exhaustive analysis. Even with a relatively large δ such as $\frac{1}{2}$, the approximation quality of TDS is excellent. One challenge for the future is to develop a mathematical analysis that explains these experimental results.

The main limitation of purely local search methods such as TDS is that they cannot always lead to globally optimal play. In particular, methods such as Hotstrat that compress information about a local game to a single number are limited in their performance. Given unlimited time, global search will eventually play optimally, while Hotstrat reaches a plateau once all temperatures are known exactly. A combination of local and global methods is necessary to further improve the performance.

TDS is applicable to any combinatorial game and is relevant for playing games that are sums of complex, hot subgames. One major future task is to apply it to Go endgames. The long term goal is to play real Go endgames at or beyond human professional level. The potential for such methods was shown in (Spight 2002), where computer-assisted human analysis uncovered several crucial errors in top-level human play.

The main technical problem in applying TDS to Go is that it must be extended to handle local *ko* repetitions. Berlekamp and Spight have developed interesting theoretical approaches to this problem (Berlekamp 1996; Spight 1999). One solution requires a coupon stack containing multiple copies of each switch. A direct implementation of these ideas will most likely be too slow in practice, so further research is needed.

Acknowledgements

This research was supported by grants from NSERC, the Natural Sciences and Engineering Research Council of Canada, and iCORE, the Province of Alberta's Informatics Circle of Research Excellence. Theodore Tegos provided his databases of Amazons positions, which were invaluable for testing our implementation of TDS.

References

Berlekamp, E.; Conway, J.; and Guy, R. 1982. *Winning Ways*. London: Academic Press. Revised version published 2001-2003 by AK Peters.

Berlekamp, E. 1996. The economist's view of combinatorial games. In Nowakowski, R., ed., *Games of No Chance: Combinatorial Games at MSRI*. Cambridge University Press. 365–405.

Kao, K. 2000. Mean and temperature search for Go endgames. *Information Sciences* 122(1):77–90.

Lieberum, J. 2003. An evaluation function for the game of Amazons. In van den Herik, J.; Iida, H.; and Heinz, E., eds., *Advances in Computer Games 10*, 299 – 308. Kluwer.

Müller, M. 1999. Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames. In *IJCAI-99*, 578–583.

Schaeffer, J. 1989. The history heuristic and the performance of alpha-beta enhancements. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11(11):1203–1212.

Spight, W. 1999. Extended thermography for multiple kos in Go. In van den Herik, H., and Iida, H., eds., *Computers and Games. Proceedings CG'98*, number 1558 in Lecture Notes in Computer Science, 232–251. Springer Verlag.

Spight, W. 2002. Go thermography - the 4/21/98 Jiang-Rui endgame. In Nowakowski, R., ed., *More Games of No Chance*. Cambridge University Press. 89–105.

Tegos, T. 2002. Shooting the last arrow. Master's thesis, University of Alberta.