A Generalized Heuristic for Can't Stop

James Glenn and **Christian Aloi**
Department of Computer Science
Loyola College in Maryland
Baltimore, Maryland, USA
{jglenn,caloi}@cs.loyola.edu

Abstract

Can't Stop is a jeopardy stochastic game played on an octagonal game board with four six-sided dice. Optimal strategies have been computed for some simplified versions of Can't Stop by employing retrograde analysis and value iteration combined with Newton's method. These computations result in databases that map game positions to optimal moves. Solving the original game, however, is infeasible with current techniques and technology. This paper describes the creation of heuristic strategies for solitaire Can't Stop by generalizing an existing heuristic and using genetic algorithms to optimize the generalized parameters. The resulting heuristics are easy to use and outperform the original heuristic by 19%. Results of the genetic algorithm are compared to the known optimal results for smaller versions of Can't Stop, and data is presented showing the relative insensitivity of the particular genetic algorithm used to the balance between reduced noise and increased population diversity.

Introduction

Can't Stop is a board game for 2-4 players invented by Sid Sackson and published by Parker Brothers in 1980 (it is currently published by Face 2 Face Games (Sackson 2007)). Can't Stop is one of a class of games called *jeopardy stochastic games* (or *jeopardy dice games* when the stochastic element is supplied by dice) in which each player's turn is a sequence of stochastic events, some of which allow the player to make progress towards a goal, and some of which will end the player's turn immediately. After each incremental step towards the goal, players can choose to end their turn, in which case the progress made during the turn is banked and cannot be lost on a later turn. Players who press their luck and choose to continue their turns risk being forced to end their turns by an adverse outcome of the stochastic event, in which case they lose any progress made during the turn. Pig, Ten Thousand, and Cosmic Wimpout are other examples of jeopardy stochastic games.

The specific rules for Can't Stop are as follows. The game is played on a board with columns labelled 2 through 12 (for the possible totals of two dice). Columns 2 and 12 are three spaces long, 3 and 11 are five spaces long, and so forth to the thirteen spaces in column 7. Each player has a set of colored markers, one for each column, with each player's markers having a color unique to that player. There are also three neutral markers (white) that are used to mark players' progress during a turn. Each turn follows these steps:

(1) the current player rolls four six-sided dice;

(2) the player groups the dice into two pairs in such a way that progress can be made in the next step – if that is impossible then the turn ends immediately with the neutral markers removed from the board and the colored markers left as they are;

(3) a neutral marker is placed one space above the player's colored marker in the column corresponding to the totals on each pair, or if there is already a neutral marker in the column for one pair then that neutral marker is advanced one space;

(4) the player chooses between returning to step (1) or ending the current turn, in which case the colored markers are moved to the position of the neutral markers.

The goal of the game is to be the first player to advance to the top of any three columns. Progress cannot be made in a column that has been won by a player. The player must use both pair totals if possible, but is allowed to choose which to use if the pairing in step (2) results in pairs such that one or the other total can be used, but both can't be used at the same time (this can happen when only one neutral marker is left).

For example, in Figure 1 the possible pair totals would be 4 and 8 or 5 and 7. In the former case the neutral marker would be moved one space up in column 8. The 4 could not be used because that column has been won. In the latter case the neutral marker in column 5 would be moved up one space and the third neutral marker would be placed at the bottom of column 7. If the roll had been 2-2-2-2 then the player would lose all progress because the only pair total that could be made would be 4 but no further progress can be made in column 4.

Solitaire Can't Stop follows the same rules. In the solitaire version of the game, the goal is to minimize the number of turns used to win the game.

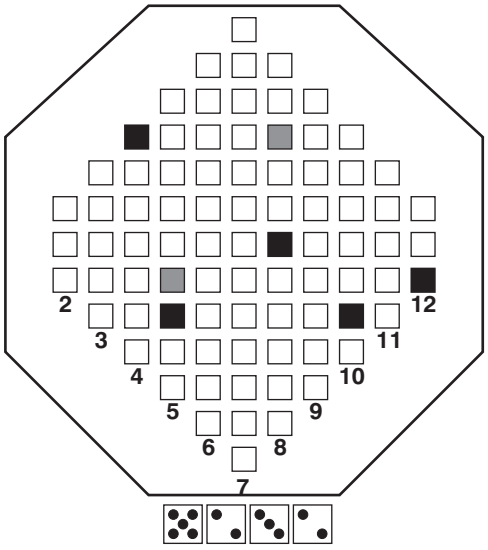

Figure 1: A Can't Stop position. Black squares represent the positions of the colored markers; gray squares are the neutral markers.

Can't Stop, Value Iteration, and Newton's Method

Retrograde analysis is a common bottom-up technique used to compute game-theoretic values of positions by starting with the terminal positions and working backwards towards the starting position (Ströhlein 1970). A simple form of retrograde analysis can be applied to acyclic games. In such cases the computation of game-theoretic values proceeds in reverse order of topological sort: as each position is examined its value can be computed based on the already-computed values of its succeeding positions. This technique has been used to solve solitaire Yahtzee (Woodward 2003; Glenn 2006).

Retrograde analysis in its more complex forms has been applied to endgames for non-stochastic games including chess (Thompson 1986; 1996), checkers (Lake, Schaeffer, & Lu 1994; Schaeffer *et al.* 2004), and Chinese chess (Wu & Beal 2001; Fang 2005a; 2005b), and has been used to solve Nine Men's Morris (Gasser 1996), Kalah (Irving, Donkers, & Uiterwijk 2000), and Awari (Romein & Bal 2003).

The cyclic and stochastic nature of Can't Stop requires a different approach. The cycles arise from the fact that a turn can end with no progress made. Value iteration is one approach to handling the cycles (Bellman 1957). The value iteration algorithm starts with estimates of the position values of each vertex. Each vertex's position value is then updated (in no particular order in the most general form) based on the estimates of its successor's values to yield a new estimated value. In this way the estimates are refined until they converge.

The structure of Can't Stop admits a refinement to this approach that uses retrograde analysis in two ways. The cycles in Can't Stop are only one turn long because progress that has been banked can never be lost (in contrast to backgammon, in which a piece that is close to being borne off can still be hit and forced back to the bar). The game graph can therefore be decomposed into components, where each component consists of an *anchor* representing the start of a turn and all of the positions that can be reached before the end of that turn.

Because the components form an acyclic graph, they are attacked in reverse order of topological sort; this is the first application of retrograde analysis. The second application is within the components: a copy of the anchor is made and all incoming edges are redirected to the copy to break the cycles within the component. An initial estimate of the anchor's position value is assigned to the copy and retrograde analysis is used to propagate position values back to the anchor. We then have the position value of the anchor as a function of the position value of the copy: $x = f(x')$. Since the position value of the copy should be the same as the position value of the original, the position value we need is the fixed point of f.

Topological value iteration, introduced by Dai & Goldsmith (2007), could find the fixed point by working backwards through each component using the position value of the anchor computed after one iteration as the estimate of the position value of the copy during the next iteration (that is, the second estimate of the copy's position value is $f(x')$, the third estimate is $f(f(x'))$, and so forth). However, in the case of Can't Stop it is possible to compute the slope $f'(x')$ which can then be used in Newton's method to compute estimates that converge more quickly to the fixed point (Glenn, Fang, & Kruskal 2008).

Heuristic Strategies

Solitaire Can't Stop has been solved for simplified variants that use dice with fewer than six-sides and a board with possibly shorter columns (Glenn, Fang, & Kruskal 2008). We will refer to these variants as (n, k) Can't Stop where n is the number of sides on the dice and k is the length of the shortest column (with adjacent columns differing by 2 in length).

The most complex version of Can't Stop that has been solved is $(5, 2)$ Can't Stop. Evaluating the 17 billion positions in that game took 60 CPU days; an estimate for the time required to solve the official game using current techniques is 3000 CPU years. Heuristic strategies for the full game are therefore still of interest. For simple games heuristics may still help human players: no human can memorize the data or mentally perform the calculations needed to replicate the optimal strategy for $(5, 2)$ Can't Stop.

The Rule of 28

One such strategy is the Rule of 28 (Keller 1986). The Rule of 28 is used to determine when to end a turn by assigning a *progress value* to each configuration of the neutral markers. Players should end their turn when this value reaches or exceeds 28. The progress value computation is split into two parts: one part for measuring the progress of the neutral markers; and one part for assessing the difficulty of making a roll that will allow further progress.

The first part of the progress value is computed as the total of the values for all the columns. The value for a column

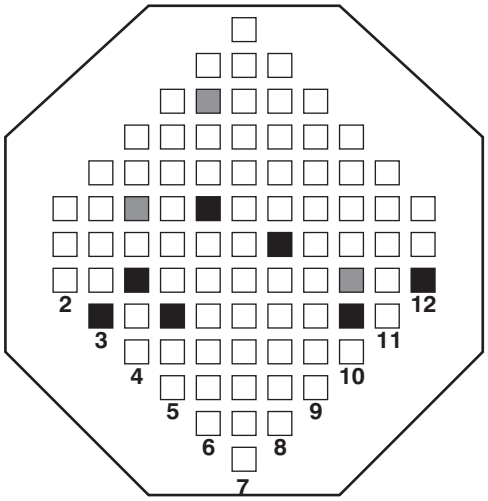

Figure 2: A position where the Rule of 28 suggests rolling again.

is equal to some constant weight assigned to that column times one more than the number of spaces advanced in that column. The weights are one for column 7, two for columns 6 and 8, and so forth to six for columns 2 and 12, reflecting the fact that it is more difficult to make progress in the outer columns, and those columns are shorter, so progress in them is therefore more valuable. If s_i is the number of spaces of progress in column i, then the total progress value is

$$\sum_{i=2}^{12}(s_i+1)(|7-i|+1).$$

Because certain combinations of columns are riskier to be in than others, a difficulty score is added to that sum. For example, if a roll is all evens then it is impossible to make an odd pair total. Therefore, two points are added to the progress value when all three neutral markers are in odd columns. On the other hand, every roll permits at least one even pair total, so if the neutral markers are all in even columns, two points are *subtracted* from the progress value. Additionally, four points are added when the columns are all high (≥ 7) or all low (≤ 7).

For example, in Figure 2 the progress value for column 4 is $(2+1)\cdot 4 = 12$, the progress value for column 6 is 8, and the progress value for column 10 is 8. Because all three neutral markers are in even columns, 2 points are subtracted to get a total progress value of $12+8+8-2 = 26$. The Rule of 28 suggests rolling again.

A similar scheme can be used to determine how to pair the dice: each column is assigned a weight and each possible move is scored according to the weights of the columns it would make progress in. The total is called the *move value*; the move with the highest move value is the one chosen. Weighting the outer columns lower than the middle columns (thus favoring choosing the middle columns) works better than the opposite pattern. In order to conserve neutral markers, a penalty is subtracted for each neutral marker used. In particular, if p_i is the number of squares advanced by a move in column i, then the total move value is

$$\sum_{c=2}^{12}(p_i(6-|7-i|)-6\,\mathrm{marker}(i))$$

where marker is a function that evaluates to 1 if the move places a new neutral marker in column i and 0 otherwise. For example, in Figure 1 using the 8 has a score of 5. Using the 5 and 7 has a score of only $4+6-6=4$, so this rule suggests using the 8.

When we henceforth refer to the Rule of 28 we mean the Rule of 28 combined with the above method of choosing how to pair the dice. This strategy averages approximately 10.74 turns to win the solitaire game.

Generalizing the Rule of 28

Any of the constants assigned to the columns can be altered, as can the threshold and any of the difficulty values. Furthermore, the spaces within a column needn't be assigned the same weights. For example, to evaluate a particular move we denote by m_i the position of the neutral marker in column i (or the colored marker if there is no neutral marker) and by n_i the position the neutral marker would advance to after the move. Assign the weight x_{ij} to space j in column i. Then the total move value v is the sum of the weight of the spaces that would be advanced over in the current turn if that move was made:

$$v=\sum_{i=2}^{12}\sum_{j=m_i+1}^{n_i}x_{ij}.$$

The same technique could be applied to progress values as well.

A further generalization could assign a difficulty score for each combination of columns individually rather than grouping them as all odds, all evens, etc. Yet another generalization could vary the weights for spaces based on the current positions of the colored markers so that near the end of the game progress is valued more in columns that are near completion than in columns that are unlikely to be finished before the game is over.

Genetic Algorithm

We used a genetic algorithm to optimize the various parameters in the generalized heuristics. The fitness of an individual strategy is taken to be the expected number of turns that strategy takes to finish the solitaire game. Two different genomes were used. There are 18 parameters encoded in the first genome: $(p_2,\ldots,p_7,v_2,\ldots,v_7,e,o,h,l,k,t)$ where

1. the p_i are the progress weights for each column (with $p_{12}=p_2$, $p_{11}=p_3$, etc.), the v_i are the move weights for each column;
2. e, o, h, and l are the difficulty scores for all even columns, all odd columns, all high columns, and all low columns respectively;
3. k is the penalty for using a marker; and
4. t is the threshold that determines when to end a turn.

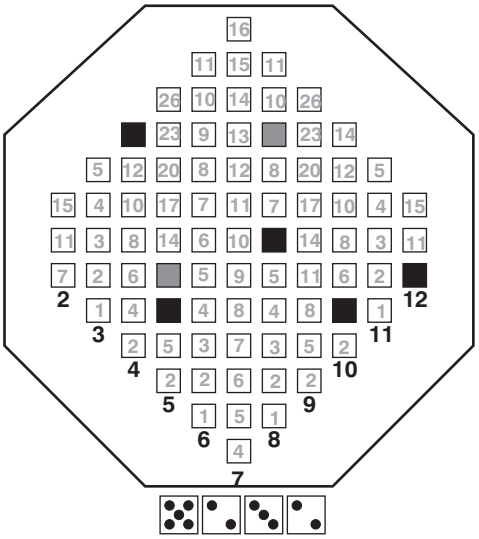

Figure 3: A Can't Stop position with linear move weights.

The column weights are encoded using three bits each (for a range of 0-7), the difficulty scores and penalties using four bits each (for a range of -8 to 7), and the threshold is encoded using five bits (for a range of 0-31), for a total of 61 bits. We will refer to this as the *Constant Weights Genome*.

Note that the weights are not normalized: a strategy with genome $(2, 2, \ldots, 2, 28)$ would behave identically to one with genome $(1, 1, \ldots, 1, 14)$. For this reason, it is expected that some of the alleles will take on their maximum possible values in order to maximize resolution.

The second genome allows the move weights to vary within a column, but not in the most general way. Instead, the weights within a column are assumed to be a linear function of the position within the column. Everything else is as in the first genome, except that $l = h$ so l is not encoded separately. This second genome is then $(p_2, \ldots, p_7, m_2, ..., m_7, b_2, ..., b_7, e, o, h, k, t)$ where the m_i and the b_i are the slopes and intercepts respectively of the linear function that determines the move weights within column i:

$$x_{ij} = \lfloor m_i \cdot \frac{j}{l_i} + b_i \rfloor$$

where l_i is the length of column i. When five bits are used to encode each slope (with the 32 possible values chosen somewhat arbitrarily from between 0 and 64) and three for each intercept (range 0-7), this genome uses 87 bits. We will refer to this as the *Linear Weights Genome*.

For example, suppose the current game position and weights for each square are as in Figure 3 with a penalty of 6 for using a neutral marker. The two possible moves are again advancing one space in column 8 or one space in both of columns 5 and 7. The move value is 10 for the first move and $14 + 4 - 6 = 12$ for the second, so the strategy would suggest making the second move.

Note that both genomes can encode the Rule of 28.

Results

The genetic algorithm was run using standard bit string operators. In particular, we use double-point crossover and children replace their parents. Each bit in the children is flipped with probability 0.04. We use two-round tournament selection to determine crossover pairs, with fitness values estimated by simulation of several games (see below for a discussion of how the accuracy of the estimations affects the results).

Comparisons

We have run the genetic algorithm using the Constant Weights Genome with many different parameters. The parameters describing the best strategy evolved can be found in Table 1. Over 250,000 games, it averaged 9.12 turns to finish, an 18% improvement over the Rule of 28. In comparison to the Rule of 28, the progress weights increase more slowly until column 3 and 2, where they are at the maximum value possible in the genome. The choice weights show a strong preference for choosing columns 2 and 12, with the odd columns completely out of favor except for column 7.

Table 1: Overall Constant Weights Champion.

Column	Progress	Move
2,12	7	7
3,11	7	0
4,10	3	2
5,9	2	0
6,8	2	4
7	1	3
Difficulty Scores		
odds		7
evens		1
highs		6
lows		5
marker		6
threshold		29

For the Linear Weights Genome the best strategy evolved achieves an average score of 9.05. Its parameters are given in Table 2. It mirrors the Constant Weights Genome champion in assigning little value to the odd columns (again, except for column 7) and very high value to the outer columns. It is interesting that the progress weights are very similar to those in the Rule of 28, suggesting that the Rule of 28 is a good strategy for determining when to roll again or stop when it is paired with a good strategy for choosing how to group the dice.

We have also run the genetic algorithm using the Linear Weights Genome for every version of Can't Stop from the official version ($(6, 3)$ Can't Stop) down to the very simplified $(2, 1)$ Can't Stop. Table 3 compares the best strategy found using the genetic algorithm to the Rule of 28 (or an analagous strategy for simplified versions) and to the optimal strategy (where available).

Finally, we have run the champions found by the genetic algorithms against each other and against the Rule of 28 in

Table 2: Overall Linear Weights Champion.

Column	Progress	Move
2,12	7	$64x + 7$
3,11	6	$24x + 1$
4,10	4	$28x + 2$
5,9	3	$8x + 1$
6,8	2	$18x + 3$
7	1	$12x + 4$
Difficulty Scores		
odds		1
evens		0
highs		-4
lows		-4
marker		4
threshold		24

Table 3: Comparison of Can't Stop Strategies.

(n, k)	Optimal	Linear Weights	Rule of N	N
$(2, 1)$	1.298	1.30	1.39	19
$(2, 2)$	1.347	1.37	1.56	29
$(2, 3)$	1.400	1.46	1.74	26
$(3, 1)$	1.480	1.60	1.87	26
$(3, 2)$	1.722	1.85	2.43	25
$(3, 3)$	1.890	2.12	2.83	30
$(4, 1)$	2.187	2.75	2.85	30
$(4, 2)$	2.454	3.06	3.45	35
$(4, 3)$	2.700	3.47	4.32	40
$(5, 1)$	2.791	4.37	5.03	26
$(5, 2)$	3.396	4.74	6.26	28
$(5, 3)$		6.15	7.45	27
$(6, 1)$		6.40	7.67	27
$(6, 2)$		7.15	9.38	28
$(6, 3)$		9.05	10.7	28

2-player games. In this setting, the strategies make decisions without considering the positions occupied by the opposing player. Probabilities of Player 1 winning are given for each combination of players in Table 4 (10,000 games simulated for each combination). Note that although the Linear Weights Genome champion performs better than the Constant Weights champion in the solitaire game, it performs worse head to head.

Effect of Noise

One challenge when using evolutionary algorithms in this context of stochastic games is that estimating the fitness values by simulation can be extremely noisy. In the case of Can't Stop we find that the standard deviation of the number of turns used by the Rule of 28 is approximately 3.25 (30% of the mean) and for the Linear Weights champion it is approximately 2.30. In addition, we suspect that the objective function is highly multi-modal and that maintaining diversity (or, in evolution strategy terms, balancing exploitation and exploration) will be essential and difficult.

Table 4: Head to Head Performance.

	P1		
P2	Rule of 28	Constant	Linear
Rule of 28	-	0.74	0.71
Constant	0.36	-	0.54
Linear	0.38	0.56	-

Arnold & Beyer (2003) find that evolution strategies are more robust than other optimization algorithms in a simple environment with high levels of noise, however, efficiency still drops with increased noise. These results clearly suggest that noise should be reduced by repeated sampling, but this must be balanced against the extra time it takes to obtain the additional samples. Fitzpatrick & Grefenstette (1988) suggest that in noisy environments and given a fixed number of function evaluations, it is better to have a larger population with fewer evaluations than a smaller population with more evaluations (and hence less noise). Glenn (2007), working in a context similar to Can't Stop (solitaire Yahtzee), presents preliminary evidence that supports that, although the improvement is in the *average* fitness of each generation; nothing about the improvement (if any) in the *best* individual is reported. However, Arnold and Beyer report that for low levels of noise efficiency drops as population size increases. Jin & Branke (2005) survey more answers to this question, along with many other approaches to dealing with noise.

We have run the genetic algorithm with different population sizes. In each case we modified the number of games simulated when estimating the fitness values so that the number of games simulated during a single generation would remain constant. The genetic algorithm was run for 20 generations; this is a few generations past where the algorithm stagnates. After the final generation, many more samples were taken to better estimate the fitnesses. The same number of samples were used during this final estimation regardless of the population size because we want equally good estimates of the best individuals' fitnesses for each run of the genetic algorithm. In Table 5 we report the mean fitness of the final population and the mean fitness of the best individual in the final population. We also report the mean fitness of the best individual among the first 100 arbitrarily chosen from the final generation. This reflects the best individual that could be found if we desired to preserve the accuracy of the final estimation yet have the number of games simulated after the final generation not vary with the population size.

It is clear that the mean fitness of the final generation decreases as the population size increases. However, it is also clear that the fitness of the best individual increases. This is not surprising, since even if the average individual in the larger populations is slightly worse, the fact that there are many more of them increases the probability of an outlier. The data for the best individual among the first 100 are less conclusive. There appears to be no significant difference

Table 5: Effect of Population Size.

Population	Samples	Final Gen.	Best of 1st 100	Overall best
100	640	10.19	9.14	9.14
125	512	10.16	9.16	9.15
160	400	10.20	9.12	9.10
200	320	10.28	9.15	9.11
250	256	10.28	9.17	9.11
320	200	10.28	9.17	9.10
400	160	10.31	9.14	9.07
800	80	10.40	9.21	9.08

between population sizes of 100 and 400. The difference between 400 and 800 is significant ($p = 0.015$); that is the only difference between adjacent rows with $p < 0.045$. It is possible (perhaps likely) that the lack of sensitivity to population size is an effect of some of the other parameters of the genetic algorithm. Further investigation is required.

Conclusion

We have generalized an existing heuristic for solitaire Can't Stop and run a genetic algorithm to optimize the parameters of the generalizations. The simpler of the two genomes yields a 18% improvement over the original heuristic. The more expressive genome yields a further 1% improvement in the average number of turns to complete the game. The trade-off between reducing noise and getting more accurate estimates of strategies' fitnesses was examined and no major effects were found in either direction for the particular genetic algorithm used.

Future work will investigate even more expressive genomes and more closely examine the effects of noise in the evaluation function.

Acknowledgments

Christian Aloi was supported by the Hauber Fellowship program in the College of Arts and Sciences at Loyola College in Maryland.

References

Arnold, D., and Beyer, H.-G. 2003. A comparison of evolution strategies with other direct search methods in the presense of noise. *Computational Optimization and Applications* 24:135–159.

Bellman, R. E. 1957. *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press.

Dai, P., and Goldsmith, J. 2007. Topological value iteration algorithm for Markov decision processes. In *International Joint Conferences on Artificial Intelligence*, 1860–1865.

Fang, H. 2005a. The nature of retrograde analysis for Chinese chess, part I. *ICGA Journal* 28(2):91–105.

Fang, H. 2005b. The nature of retrograde analysis for Chinese chess, part II. *ICGA Journal* 28(3):140–152.

Fitzpatrick, J., and Grefenstette, J. 1988. Genetic algorithms in noisy environments. *Machine Learning* 3:101–120.

Gasser, R. 1996. Solving nine men's Morris. *Computational Intelligence* 12:24–41.

Glenn, J.; Fang, H.; and Kruskal, C. P. 2008. Retrograde approximate algorithms for some stochastic games. *ICGA Journal* 31(2):77–96.

Glenn, J. 2006. An optimal strategy for Yahtzee. Technical Report CS-TR-0002, Loyola College in Maryland, 4501 N. Charles St, Baltimore MD 21210, USA.

Glenn, J. 2007. Computer strategies for solitaire Yahtzee. In *IEEE Symposium on Computational Intelligence and Games (CIG 2007)*. 132–139.

Irving, G.; Donkers, J.; and Uiterwijk, J. 2000. Solving Kalah. *ICGA Journal* 23(3):139–147.

Jin, Y., and Branke, J. 2005. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation* 9(3):303–317.

Keller, M. 1986. Can't stop? Try the rule of 28. *World Game Review* 6. See also http://www.solitairelaboratory.com/cantstop.html last visited Nov. 22, 2008.

Lake, R.; Schaeffer, J.; and Lu, P. 1994. Solving large retrograde analysis problems using a network of workstations. In van den Herik, H.; Herschberg, I. S.; and Uiterwijk, J., eds., *Advances in Computer Games VII*. Maastricht. the Netherlands: University of Limburg. 135–162.

Romein, J., and Bal, H. 2003. Solving the game of Awari using parallel retrograde analysis. *IEEE Computer* 36(10):26–33.

Sackson, S. 2007. *Can't Stop*. Providence, RI, USA: Face 2 Face Games. Boxed game set.

Schaeffer, J.; Björnsson, Y.; Burch, N.; Lake, R.; Lu, P.; and Sutphen, S. 2004. Building the checkers 10-piece endgame databases. In van den Herik, H.; Iida, H.; and Heinz, E., eds., *Advances in Computer Games 10. Many Games, Many Challenges*. Boston, USA: Kluwer Academic Publishers. 193–210.

Ströhlein, T. 1970. *Untersuchungen über kombinatorische Spiele*. Ph.D. Dissertation, Fakultät für Allegemeine Wissenschaften der Technischen Hochschule München, Munich.

Thompson, K. 1986. Retrograde analysis of certain endgames. *ICCA Journal* 9(3):131–139.

Thompson, K. 1996. 6-piece endgames. *ICCA Journal* 19(4):215–226.

Woodward, P. 2003. Yahtzee: The solution. *Chance* 16(1):18–22.

Wu, R., and Beal, D. 2001. Fast, memory-efficient retrograde algorithms. *ICGA Journal* 24(3):147–159.