NOTES

AWARI IS SOLVED

John W. Romein[1] *and Henri E. Bal*[2]

Amsterdam, The Netherlands

1. THE RESULT

The main message of this note is that we solved awari. With perfect play, the game ends in a draw.

Awari (also known as wari, owari, awale, awele, and ayo) is an ancient and well-known mancala variant that originates from Africa, and is played worldwide now. The rules are given in the Appendix. Although the rules are simple, the game requires profound strategic insight to be played well.

We solved the game by searching the entire state space on a parallel computer with 144 processors. The state space contains *889.063.398.406* positions, and is searched using retrograde analysis. The analysis determined the best moves and exact outcomes for all reachable board positions, starting in the final positions, and searching backward to the initial position. No single computer has enough processing power and memory to search the state space, but even on a modern, parallel computer the problem was extremely challenging.

For each number of stones on the board, a separate database was computed that contains the eventual outcomes (after perfect play) of the n-stone positions. The hardest database was the 48-stone database, which contains over 200 billion positions, larger than any (endgame) database computed for whichever game so far. Others tried to solve the game by creating endgame databases of up to 40 stones and perform a forward search from the root, but were unsuccessful because the forward search did not hit the endgame databases fast enough. We therefore computed *all* databases of up to 48 stones (except the 47-stone database, since no 47-stone positions are reachable) by retrograde analysis.

Retrograde analysis requires much main memory, due to the random accesses of database entries during construction. The choice or design of the algorithm to create awari (endgame) databases is mainly determined by the amount of main memory available (Lincke, 2002), trading memory for additional computational effort and storing intermediate results on disk. Our system contains 144 Pentium III processors at 1.0 GHz, 72 GB of distributed main memory, a total disk space of 1.4 TB, and a Myrinet interconnect: a fast, switched network. One of the challenges was to handle the relative “small” amount of memory. The parallel retrograde search algorithm described by Bal and Allis (1995) is efficient, but would have required more than 350 GB of memory, much more than we had. Sequential memory-limited search algorithms for awari endgame databases exist (cf. Lincke and Marzetta, 2000), but solving awari entirely on a single machine would take decades, if not centuries, since these algorithms still require much more memory than a single computer provides. We developed an efficient parallel algorithm that partitions the work and the database entries over the machines, and uses the memories economically.

Another challenge was to handle the interprocessor communication efficiently. The random accesses of database entries result in many messages being sent between the processors. Our algorithm makes all communication asynchronous by migrating *work* instead of *data*, hiding network latency and keeping the processors busy. If processor A needs data from B for some computation, it does not ask B for the data (which requires waiting for a reply message), but sends the work to B (which is done asynchronously), and continues doing other work. The power of this idea was already demonstrated in forward search (Romein *et al.*, 2002). Our

[1]Vrije Universiteit, Faculty of Sciences, Amsterdam, The Netherlands. Email: john@cs.vu.nl
[2]Vrije Universiteit, Faculty of Sciences, Amsterdam, The Netherlands. Email: bal@cs.vu.nl

algorithm balances and overlaps the use of all resources: processors, disks, memory, and network. Despite the high network bandwidth requirements (we exchanged 1.0 petabit (= 10^{15} bits) of data!), the algorithm is efficient: it required only 51 hours to solve the game. The details of the algorithm, statistics on the computed databases, and a summary of the measures to verify the databases are described separately (Romein and Bal, 2002).

The fact that awari is a draw makes it a fine game: it is neither an advantage, nor a disadvantage to begin the game. It was not a surprise that awari is a draw; however, there are quite some misconceptions about good openings (see also the next section). For example, it was expected that 1. F4 f4 would be the only non-losing opening, but 1. F4 b4 and 1. F4 e4 lead to draws as well. Other openings, believed to be draws, are now rejected since they lose.

We set up a webserver with a database that contains the computed results for all reachable awari positions. The database is 778 GB large. Via a Java applet, the best move(s) and eventual score for all positions can be looked up. The applet can also be used to play a game against the database. By default, we decreased the playing strength by allowing the computer to make (small) mistakes, so that the human can actually win a game. After all, being defeated by a program from which one *cannot* actually win, may not be satisfactory. The applet and more statistics can be accessed via `http://awari.cs.vu.nl/`.

Did we ruin a perfectly fine game? We do not think so. Connect-4 was solved as well, and people still play the game. The same holds for other solved games. At professional level, things may change. On the one hand, the database may help improving the understanding of the game. On the other hand, it is not unlikely that the game will be excluded from the Mind Sports Olympiad (even though they use slightly different rules).

2. TOURNAMENT GAMES REANALYZED

With the perfect knowledge from our database, we can analyze the games played by world-champion-level programs. In the September issue of the *ICGA Journal*, Lincke and Van der Goot (2000) reported on the games played at the 2000 Computer Olympiad in London. Competitors were the programs Softwari and Marvin. It was expected that the programs would play nearly optimal, and therefore it was already surprising that three out of eight games actually had a winner. Marvin won the tournament by 4.5–3.5.

Both programs used an opening book and complete endgame databases for up to 34 stones. Because of the endgame databases, the games were ended as soon as there were 34 or fewer stones on the board, since the programs can play the remainder of the game perfectly.

Below, we annotate the games from the tournament. The original text (Lincke and van der Goot, 2000) is copied in a plain font; our annotations are displayed using bold, italic characters. Wrong moves (i.e., moves that lead to fewer captures than possible) are highlighted in inverse colours. The correct move(s) are appended in subscript, between brackets. The superscript numbers show the score for the first player from that moment on; a score of +4 means that the first player can win the game with 26–22 if no further mistakes are made. Subsequent wrong moves change the score again. For example, d1$^{+1}_{[a11,e9]}$ means that the program played d1, while it should have played a11 or e9; the first player then has an advantage of 1 stone. For each program, the last move from the opening book is marked by a *.

Two games (3 and 4) were played perfectly, and quickly drawn in about ten moves. The remaining games became more interesting as they required more moves. In these games, the programs made more errors, even while in their opening books, and sometimes as soon as on the third move! In the final and decisive game, Marvin did 13 wrong moves, but still won the game! In games 1, 5, and 7, both programs have been in a winning position; in games 2, 6, and 8, the program that was behind (repeatedly) drew level with the opponent. The winning chances changed mutually, but the competitors were not always aware of this.

Although the programs made more errors than expected, they did not play badly. Marvin did the right move in 82% of the cases, and Softwari in 87% of the cases. On average, Marvin lost 0.27 points per move, and Softwari 0.25 points (losing a point means: giving the opponent the opportunity to capture one stone more). At this rate, we estimate that the programs would lose with about 27–21 when playing against our database.

3. THE GAMESCORES ANNOTATED

Game 1: MARVIN–SOFTWARI [1–0]
1. F4 f4 2. B5 d5 3. B1 f1 4. F1 b5 5. D6×2 f1 6. F1 b1 7. C8 c9×2 8. D2 b1 9. F1 d2 10. B1 c1 11. C2 d1$^{+1}_{[a11,e9]}$ 12. D1 e10 13. C1 c1* 14. E12$^{0}_{[A11]}$ c1 15. C1 f4 16. C1$^{-2}_{[D5]}$ e1 17. B3 f1$^{0}_{[d3]}$ 18. C1* d3 19. D8×4 d1 20. A15 c2 21. B2 f1 22. D3 d2??$^{+2}_{[e3]}$ [e3 is a draw] 23. A1 b4 24. E5×4 f2×2 25. C3 a17×3 [25–23]

Game 2: SOFTWARI–MARVIN [1–0]
1. F4 f4 2. B5 d5 3. B1 f1 4. E5 d1 5. C8 a8$^{+2}_{[d1]}$ 6. E1 c8 7. B2$^{0}_{[A9]}$ d3 8. C3 b8 9. C1 d1 10. B1 c1 11. C1 d1 12. A11 c1 13. D14×3 d3×2 14. F8 d1$^{+1}_{[c2]}$ 15. C2 a1$^{+2}_{[c2]}$ 16. D1 e18×2* 17. D2 a1 18. C2 d1 19. D1$^{+1}_{[F1]}$ e1 20. F1 a1 21. B5 f11 22. C2 a2$^{+2}_{[b8]}$ 23. E15 d2 24. B2 a2 25. D5* a1 26. E1 f2 27. B1 e3 28. F6×2$^{0}_{[A7]}$ a1??$^{+1}_{[e1]}$ [a1 is a loss, e1 leads to a balanced position] 29. C3 e1 30. A7 a1 31. F2 a1 32. D2 d1$^{+3}_{[b20]}$ 33. C1 f1 34. F1 e1$^{+4}_{[a1]}$ 35. D1 c10$^{+10}_{[f1×2]}$ 36. F1×3 f2$^{+12}_{[b20]}$ 37. C1$^{+10}_{[D1]}$ e1 38. D2 f1$^{+11}_{[b20]}$ 39. B4 b20 40. B2 c2 41. E9 a2 42. D4×3 d5 [29.5–18.5]

Game 3: MARVIN–SOFTWARI [1/2–1/2]
1. F4 f4 2. B5 c5 3. F1 d6 4. E5 d1 5. B2 c1 6. D8×3 d2* 7. E1 c1 8. C8 a10×4 9. A8×3* f2×3 10. A1 b10 11. F6×2 [24–24]

Game 4: SOFTWARI–MARVIN [1/2–1/2]
1. F4 f4 2. B5 c5 3. E5 b6×2 4. C6×3 b1 5. F3×2 e6×2 6. C1 b1 7. B1 f3×2 8. B1 a9 9. C2 c2 10. A10×3 [24–24]

Game 5: MARVIN–SOFTWARI [1/2–1/2]
1. F4 f4 2. E4 a6 3. A6$^{-1}_{[C5]}$ b7 4. E1 a1 5. C7 b2*$^{0}_{[e6×2]}$ 6. E1 a1$^{+2}_{[d9]}$ 7. D7$^{0}_{[B7]}$ b2 8. B7$^{-2}_{[E1]}$ d11 9. C2 b2 10. F8*$^{-3}_{[E4]}$ b1 11. D3 d2 12. A3 a5$^{+1}_{[c17×3]}$ 13. C1 f6×2 14. B4 c18$^{+2}_{[b1,d1]}$ 15. F2×5 f2×3 16. A4$^{0}_{[C4]}$ e13×2 17. D8×3 [24–24]

Game 6: SOFTWARI–MARVIN [1/2–1/2]
1. F4 f4 2. E4 a6 3. C5 e5 4. E1 c7 5. A8$^{-2}_{[B7]}$ c1$^{-1}_{[d7×2]}$ 6. E1 e1 7. D9* c1 8. E1 a3 9. C3 f5×4* 10. B9×2 c2 11. E1 e1 12. A2 a1$^{0}_{[b12]}$ 13. C3 f1 14. A1 d13 15. E2×2 e2×2 16. C1 f3 17. C1 c1 18. B4 d1 19. A1 b14 20. F15 c4×3 21. E3×3 [24–24]

Game 7: MARVIN–SOFTWARI [1/2–1/2]
1. F4 f4 2. C5 c5 3. B6 b7×2 4. A7 b1 5. F3×3 f2×3 6. C1$^{-1}_{[A1]}$ d7 7. A2 b1*$^{0}_{[a9]}$ 8. C2 a9 9. C1 d1 10. B3 c2 11. A1 f2×2 12. C1 e10 13. A2 d1 14. F1×2* b2 15. D16 a2 16. A1 f2 17. F2 a1 18. A1 e2 19. A1 d2$^{+2}_{[f1]}$ 20. B7 a1 21. D1$^{0}_{[F1]}$ e1 22. C4 b7 23. D1 d1 24. B1 a1 25. A1 c7 26. F2×2 [24–24]

Game 8: SOFTWARI–MARVIN [0–1]
1. F4 f4 2. C5 c5 3. A6$^{-2}_{[B6,F1]}$ d6$^{-1}_{[b6]}$ 4. B8 c1 5. C3* b7* 6. E8?$^{-3}_{[C1]}$ [C1 is better, after this move SOFTWARI was at least two stones down for the rest of the game ***Actually,* SOFTWARI *drew level with* MARVIN *three times in the remainder of the game***] 6. … d4×2 7. D10 d1$^{-2}_{[a10×2]}$ 8. E1 c3 9. C1 e12

10. C1 d2 11. B2 c1 12. C1 e1 13. E1 b3$^{-1}_{[d1]}$ 14. D5×2 e1? [d2 is better ***No! d2 loses one point***] 15. A6 d2?$^{0}_{[b1]}$ [b1 is better] 16. D1?$^{-1}_{[F11]}$ [after the last two bad moves of MARVIN, SOFTWARI had the chance to move into a balanced position with F11] 16. . . . e1$^{0}_{[b1]}$ 17. B1 b1 18. C2$^{-2}_{[F11]}$ c1 19. F11 d2 20. A1 c1$^{-1}_{[e2]}$ 21. D2 e2 22. F1$^{-2}_{[C1]}$ d1$^{0}_{[b1,f17\times2]}$ 23. C1 f17 24. D3$^{-3}_{[C2]}$ e2 25. C2$^{-4}_{[F1]}$ d1 26. F1 e1 27. A4 c1 28. D2 b2 29. C1$^{-5}_{[F1]}$ d2 30. F1 e1 31. E12$^{-6}_{[B5]}$ b1 32. B6 a21×3 33. A3 f7$^{-5}_{[d3,e3]}$ 34. A1 d3$^{-3}_{[c5]}$ 35. A1$^{-6}_{[C6]}$ a1$^{-5}_{[f1]}$ 36. C6 c6 37. D8×3 e6$^{-2}_{[b6]}$ 38. A2 d2 39. d1$^{-6}_{[E4\times2]}$ f2$^{-5}_{[b6,e1]}$ 40. A1$^{-6}_{[E5,F7]}$ e1 41. F7 d1$^{-4}_{[b7]}$ 42. A1 f2 43. A1 e2 44. A1 c2 45. E5×2 [22–26]

4. REFERENCES

Allis, L. V., Meulen, M. van der, and Herik, H. J. van den (1991). Databases in Awari. *Heuristic Programming in Artificial Intelligence 2: the Second Computer Olympiad* (eds. D. N. L. Levy and D. F. Beal), pp. 73–86. Ellis Horwood, Chichester, England.

Bal, H. E. and Allis, L. V. (1995). Parallel Retrograde Analysis on a Distributed System. *Supercomputing '95*, San Diego, CA.

Lincke, T. R. (2002). *Exploring the Computational Limits of Large Exhaustive Search Problems.* Ph.D. thesis, ETH Zurich, Swiss.

Lincke, T. R. and Goot, R. van der (2000). Marvin Wins Awari Tournament. *Journal of the International Computer Games Association*, Vol. 23, No. 3, pp. 173–174.

Lincke, T. R. and Marzetta, A. (2000). Large Endgame Databases with Limited Memory Space. *Journal of the International Computer Games Association*, Vol. 23, No. 3, pp. 131–138.

Romein, J. W. and Bal, H. E. (2002). Solving the Game of Awari using Parallel Retrograde Analysis. *Submitted for publication*.

Romein, J. W., Bal, H. E., Schaeffer, J., and Plaat, A. (2002). A Performance Analysis of Transposition-Table-Driven Scheduling in Distributed Search. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 5, pp. 447–459.

5. APPENDIX: RULES OF AWARI

Awari is played on a board that contains two rows of six pits, in which stones are kept. Each player owns one of the rows. The game starts with four stones in each pit. When a player is to move, he chooses one of his own nonempty pits, takes all stones from it, and sows them one by one, counterclockwise over the remaining pits, skipping the emptied pit if there are more than 11 stones. If the last stone ends in an opponent's pit and contains two or three stones after sowing, they are captured; if the second last pit holds the same conditions, they are captured as well, and so on. The objective of the game is to capture more than 24 stones. The player must give the opponent a countermove, unless no such move is available. If a player cannot move, the remaining stones are captured by the opponent. A repeated position leads to an even division of the remaining stones, ending the game.

Unfortunately, the rules slightly differ from one country to another. For example, the "grand slam rule", which states what happens if all opponent's pieces are captured in a single move (if allowed at all !) comes in many variants. We use the rules that are used by all computer scientists since the first publication on computer-aided play of awari (Allis, van der Meulen, and van den Herik, 1991).