https://dx.doi.org/10.22161/ijaers.6.4.43 ISSN: 2349-6495(P) | 2456-1908(O)

Two Classic Chess Problems Solved by Answer Set Programming

Warley Gramacho da Silva[1], Tiago da Silva Almeida[1], Rafael Lima de Carvallho[1], Edeilson Milhomem da Silva[1], Ary Henrique de Oliveira[1], Glenda Michele Botelho[1], Glêndara Aparecida de Souza Martins[2]

[1]Department of Computer Science, Federal University of Tocantins, BRAZIL
[2]Laboratory of Kinetics and Modeling Process, Department of Food Engineering, Federal University of Tocantins, BRAZIL

Abstract*—The n-Queen and the Knight's tour problem are studied by several authors who have proposed some methods to solve them. The ASP (Answer Set Programming) is a form of declarative programming oriented to difficult search problems; however, the literature does not present its use in solving these two classic and interesting chess puzzles. Thus, this work aims to solve the n-Queen and Knight's Tour problems by ASP and show it can solve combinatorial problems.*

***Keywords*— Combinatorial problems, Answer Set Programming, Computation applied.**

I. INTRODUCTION

Answer Set Programming (ASP), is a form of declarative programming oriented to a difficult search problem, mainly NP-hard [16, 17, 18].

The ASP has application in relevant industrial projects because to the availability of some efficient ASP systems [16, 17]. Nevertheless, ASP can be applied to several areas of science and technology, for example: automated product configuration, decision support for space shuttle and automatic route search [17].

The classical problems involving chess are a constant subject of heuristic and optimization studies [10,12,13]. The n-Queen Problem consists of finding the position of n-queens on a chessboard n x n. The Knight's Tour Problem aims to construct a sequence of admissible moves made by a chess knight from one square to another so that they land on each square of a board exactly once.

Both problems are interesting classical chess puzzles solved by many computational and mathematical methods [9, 11].

In this sense, the objective of this work is to propose the solution of these two classic challenges of chess through Answer Set Programming (ASP), proving that ASP is able to solve combinatorial problems.

II. CHESS PROBLEMS DESCRIPTION

a. The Knight's Tour Problem

The knight's tour problem consists of a series of moves (in an L-shape, see Fig. 1) made by a knight visiting every square of an n x n chessboard exactly once [14, 15, 19]. We can define the problem as knight's graph for n x n chessboard to be graph $G = (V, E)$ where $V = \{(i,j)|1 \leq i,j \leq n\}$, and $E = \{((i,j),(k,l))|\{|i-k|,|i-l|\} = \{1,2\}\}$. Such that, there is a vertex for every square of the board and an edge between two vertices exactly when there is a knight move from one to another. A knight's tour is called closed if the last square visited is also reachable from the first square by a knights move, i.e., an open knight's tour is defined to be a Hamiltonian path; and open otherwise, i.e., closed knight's tour is defined to be Hamiltonian cycle on a knight's graph [19].

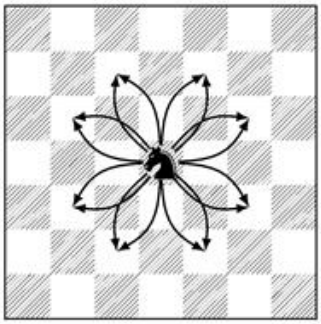

Fig. 1: Knight possible moves in an L-shape

The knight's tour problem is used as the basis of studies for the development of cryptographic schemes [14] and implementation of random binary numbers [15].

The literature points to some methods that propose the solution of the knight's tour problem, such as Artificial Bee Colony [10] and structural algorithms with pre-defined heuristic rules [20].

b. The *n*-Queens Problem

The *n*-Queens problem is to place n queens (a queen can move as far as she pleases, horizontally, vertically, or diagonally. See Fig. 2), on an $n \times n$ chessboard in such a way that no queen can attack another, i.e., so that no two queens are placed in the same row or column or on the same diagonal. This problem is a generalization of the

https://dx.doi.org/10.22161/ijaers.6.4.43 *ISSN: 2349-6495(P) | 2456-1908(O)*

original 8-Queen's problem [7]. Survey of known results is given in [1].

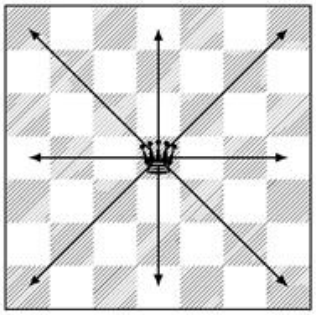

Fig. 2: Queen piece available moves

Let $K = \{0,1,\ldots,p-1\}$, we can uniquely assign to each position on the board a par (i,j) of coordinates in the usual manner, with $i,j \in K$. Then a solution can be thought of as a permutation f from K to K satisfying (1) e (2) for all x, y in K, $x \neq y$

(1) $f(x) - x \neq f(y) - y$

(2) $f(x) + x \neq f(y) + y$

Such permutation f will be called ordinary solution. Instead of condition (1) and (2) one might also consider permutations f satisfying (a) and (b) for all x, y in K, $x \neq y$

(a) $f(x) - x \neq f(y) - y \ (modp)$

(b) $f(x) + x \neq f(y) + y \ (modp)$

A permutation f satisfying (a) and (b) is called modular solution. Any modular solution is also an ordinary solution.

The *n*-Queens problem is often studied because there are several practical applications: VLSI (Very Large Scale Integration) testing, traffic control, parallel memory storage schemes, and deadlock prevention [6, 5], memory storage scheme for conflict free access for parallel memory systems [2,3,4].

III. IMPLEMENTATION AND EXPERIMENTAL RESULTS

All the experiments presented in this section have been performed with CLINGO 4.5.3.

The CLINGO program shown in Listing 1 solves the open knight's tours problem. In the Listing 1, on line 1 and 2 we define the chessboard and line 3 defines the number of step. On line 5 expresses that at step I there can be one and only one position. Line 6-9 to force the next steps to execute the knight's tours rule first, we give the definition of next, then say that there can be no steps without the rule being verified and finally we say that you can not go back to the same cell twice. Line 10-14 next steps are related to the rule of the horse and return to the same cell. Line 15 defines the starting position chessboard where the knight's will start.

```
xchessboard(1..m).
ychessboard(1..n).
time(1..m*n).
xypos(X,Y) :- xchessboard(X), ychessboard(Y).
1 { position(I,X,Y) : xypos(X,Y) } 1 :- time(I).
fromTO(XO,YO,XT,YT) :- xypos(XO,YO),
xypos(XT,YT), |XO-XT| = 1, |YO-YT| = 2.
fromTO(XO,YO,XT,YT) :- xypos(XO,YO),
xypos(XT,YT), |XO-XT| = 2, |YO-YT| = 1.
:- time(I), time(I+1), xypos(XO,YO), xypos(XT,YT),
   position(I,XO,YO), position(I+1,XT,YT), not
fromTO(XO,YO,XT,YT).
:- time(IA), time(IB), IA < IB, xypos(X,Y),
position(IA,X,Y), position(IB,X,Y).
:- position(1,X,Y), X+Y>2.
```

Listing 1: Open knight's tours program.

The CLINGO program shown in Listing 2 solves the closed knight's tours problem. The ideal is the same as the open knight's tours program, the difference between listing 1 and 2 are in line 3 which defines an additional step of the knight which is the return of the knight initial position after visiting all the cells and in line 16 forces that return.

```
xchessboard(1..n).
ychessboard(1..m).
time(1..n*m+1).
xypos(X,Y) :- xchessboard(X), ychessboard(Y).
fromTO(X1,Y1,X2,Y2) :- xypos(X1,Y1), xypos(X2,Y2),
|X1-X2| = 1, |Y1-Y2| = 2.
fromTO(X1,Y1,X2,Y2) :- xypos(X1,Y1), xypos(X2,Y2),
|X1-X2| = 2, |Y1-Y2| = 1.
1 { position(I,X,Y) : xypos(X,Y) } 1 :- time(I).
:- time(I), time(I+1), xypos(X1,Y1), xypos(X2,Y2),
   position(I,X1,Y1), position(I+1,X2,Y2), not
fromTO(X1,Y1,X2,Y2).
:- time(I1-1), time(I2), I1 < I2, xypos(X,Y),
   position(I1,X,Y), position(I2,X,Y).
:- position(1,X,Y), X+Y>2.
:- position(n*m+1,X,Y), X+Y>2.
```

Listing 2: Closed knight's tours program.

Fig. 3-6 show solutions for the open Knight's Tour problem on chessboard (5x5), (6x6), (8x8), and (6×5), respectively.

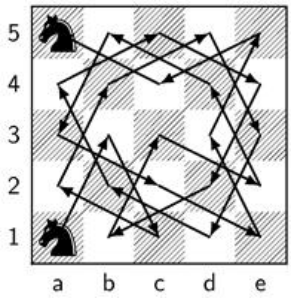

Fig. 3: Open Knight's Tour on chessboard (5×5).

https://dx.doi.org/10.22161/ijaers.6.4.43 ISSN: 2349-6495(P) | 2456-1908(O)

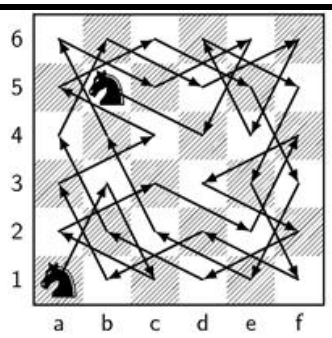

Fig. 4: Open Knight's Tour on chessboard (6×6).

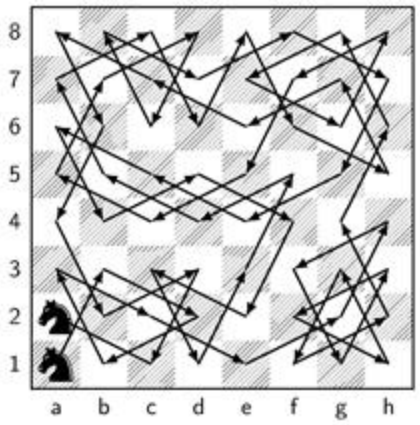

Fig. 5: Open Knight's Tour on chessboard (8×8).

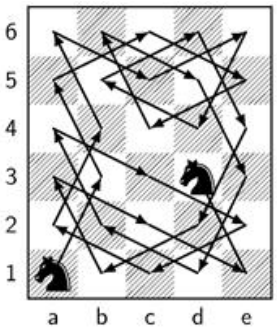

Fig. 6: Open Knight's Tour on chessboard (6×5).

Solutions for the closed *Knight's Tour* are show in Fig. 7-9 for chessboard (6x6), (8x8), and (6×5), respectively.

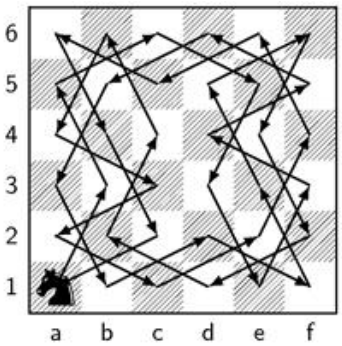

Fig. 7: Closed Knight's Tour on chessboard (6×6).

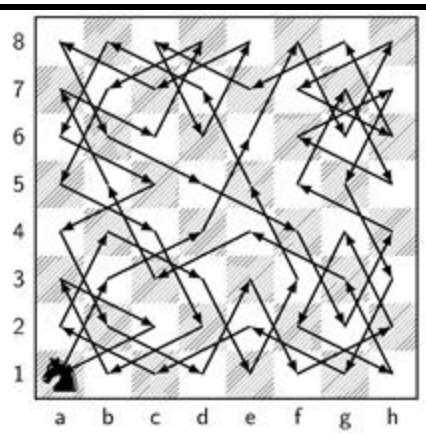

Fig. 8: Closed Knight's Tour on chessboard (8×8).

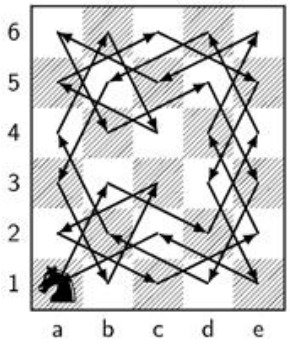

Fig. 9: Closed Knight's Tour on chessboard (6×5).

The CLINGO program shown in Listing 3 solves the *n*-Queens Problem. In the Listing 3, on line 1 and 2 we place queens on the chess board exactly one queen per row/column; on line 3 and 4 allows at most one queen per diagonal.

```
1 { queen(I,1..n) } 1 :- I = 1..n.
1 { queen(1..n,J) } 1 :- J = 1..n.
:- 2 { queen(D-J,J) }, D = 2..2*n.
:- 2 { queen(D+J,J) }, D = 1-n..n-1.
```

Listing 3: n-Queens program.

Fig. 10-13 show solutions for 5x5, 6x6, 7x7, and 8x8 n-queen's problems respectively.

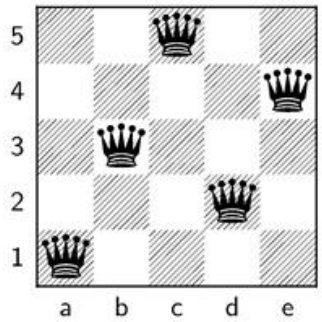

Fig. 10: 5×5 Queen's solution

https://dx.doi.org/10.22161/ijaers.6.4.43 *ISSN: 2349-6495(P) | 2456-1908(O)*

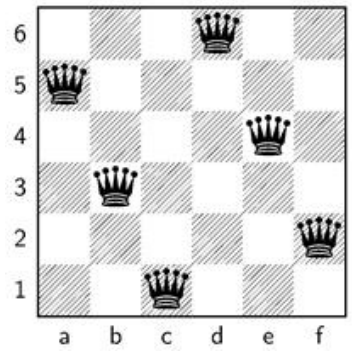

Fig. 11: 6×6 Queen's solution

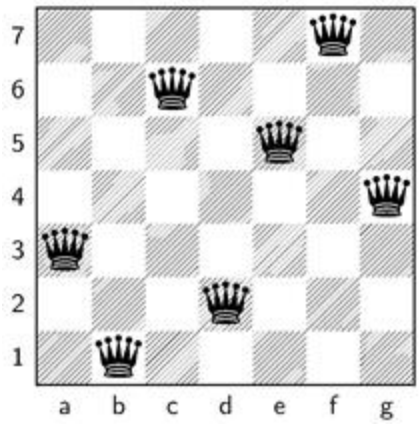

Fig. 12: 7×7 Queen's solution

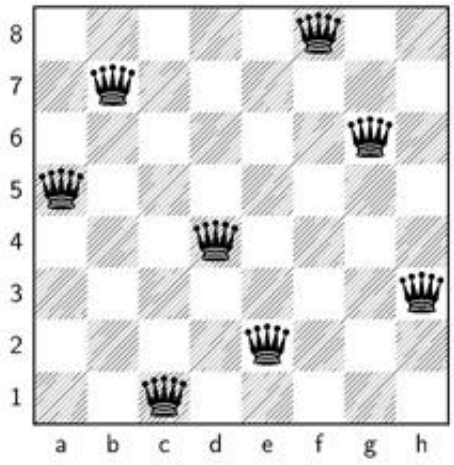

Fig. 13: 8×8 Queen's solution

IV. CONCLUSION

This paper presents the solution to two classic chess problems (Knight's Tour and *n*-Queens) through the use of ASP.

We have seen an ASP algorithm for constructing closed and open Knight's Tours on square boards (5x5, 6x6, and 8x8) and not square boards (6x5).

We also present solutions to the problem of *n*-Queens on square boards 5x5, 6x6, 7x7, and 8x8, proving that the ASP algorithm is able to solve combinatorial problems.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the Applied Computing Team (NCA) and of the Laboratory of Kinetics and Modeling Process (LaCiMp) for support.

REFERENCES

[1] Jordan Bell and Brett Stevens. A survey of known results and research areas for n-queens. Discrete Mathematics, 309(1):1– 31, 2009.

[2] C. Erbas, M. M. Tanik, and V. S. S. Nair. A circulant matrix based approach to storage schemes for parallel memory systems. In Proceedings of 1993 5th IEEE Symposium on Parallel and Distributed Processing, pages 92–99, 1993.

[3] C. Erbas and M.M. Tanik. n-queens problem and its algorithms. Technical Report 91-CSE-8, Department of Computer Science and Engineering, Southern Methodist University, 1991.

[4] C. Erbas and M.M. Tanik. Storage schemes for parallel memory systems and then-queens problem. In Proceedings of the 15th Anniversary of the ASME ETCE Conference, Computer Applications Symposium, volume 43, pages 115–120, 1992.

[5] Cengiz Erbas, Murat M. Tanik, and ZekeriyaAliyazicioglu. Linear congruence equations for the solutions of the N-queens problem. Information Processing Letters, 41(6):301–306, 1992

[6] RokSosic and Jun Gu. A polynomial time algorithm for the n-queens problem. SIGART Bull., 1(3):7–11, October 1990.

[7] ZsuzsannaSzaniszlo, Maggy Tomova, and Cindy Wyels. The n-queens problem on a symmetric toeplitz matrix. Discrete Mathematics, 309(4):969 – 974, 2009.

[8] Rodney W. Topor. Fundamental solutions of the eight queens problem. BIT Numerical Mathematics, 22(1):42–52, Mar1982

[9] Farhan, A.S.; Tareq, W.Z.; Awad, F.H. Solving N Queen Problem using Genetic Algotihm. International Journal of Computer Applications. Vol. 122, N. 12, 2015.

[10] Banharnasakun, A. Artificial Bee Colon Algorithm for solving the Knight's. In: International Conference on Intelligent Computing & Optimization. Pp. 129-128. 2018.

[11] Buño, K.C.; Cabarle, F.G.C.; Calabia, M.D.; Adorna, H.N. Solving the N-Queens problem using Dp systems with active membranes. Theoretical Computer Science. Vol. 736, p.1-14, 2018.

[12] Torggler, V.; Aumann, P.; Ritsch, H.; Lechner, W. A Quantum N-Queens Solver. Disponível em https://arxiv.org/pdf/1803.00735.pdf . Acessoem 17/02/2018.

https://dx.doi.org/10.22161/ijaers.6.4.43 *ISSN: 2349-6495(P) | 2456-1908(O)*

[13] Muhammad, K.; Gao, S.; Qaisar, S.; Abdul, M.M. Muhammad, A.; Usman, A.; Aleena, A.; Shahid. Comparative Analysis of Meta-Heuristic Algorithms for Solving Optimization Problems. In: 8th International Conference on Management, Education and Information. 2018.

[14] Singh, M.; Kakkar, A.; Singh, M. Image Encryption Scheme Based on Knight's Tour Problem. In: 4th International Conference on Eco-friendly Computing and Communication Systems

[15] Mahamood, A.S.; Rahim, M.S.M.; Othman, N.Z.S. Implementation of the Binary Random Number Generator Using the Knight Tour Problem. Modern Applied Science. Vol. 10, n.4, 2016.

[16] BONATTI, P. et al. A 25-year perspective on logic programming. In: DOVIER, A.; PONTELLI, E. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2010. cap. Answer Set Programming, p. 159–182.

[17] LIFSCHITZ, V. Answer set programming and plan generation. ARTIFICIAL INTELLIGENCE, v. 138, p. 2002, 2002.

[18] EITER, T.; IANNI, G.; KRENNWALLNER, T. Answer set programming: A primer. In: TESSARIS, S. et al. (Ed.). Reasoning Web. [S.l.]: Springer, 2009. (Lecture Notes in Computer Science, v. 5689), p. 40–110.

[19] Ian Parberry. An efficient algorithm for the knightstour problem. Discrete Applied Mathematics, pages 251–260,1997.

[20] Tan Chi Wee, Nur Hafizah Ghazali and Ghazali Bin Sulong, A New Structured Knight Tour Algorithm by Four Knights with Heuristic Preset Rules. Journal of Telecommunication, Electronic and Computer Engineering, 171-175, 2018.