LIÈGE université
Sciences Appliquées

INFO9015-1 - Logic for Computer Science

Project Report - Knight's Tour SAT solving

Teacher:
Pascal Fontaine

Author:
Valérian Wislez s200825

Teaching Assistant:
Tom Clara

Academic Year 2025-2026

1 Introduction

This paper describes the way each question of the project has been solved. A more comprehensive view on the submitted source code can be found at [3].

2 Naive solution

The first attempt at solving the problem involves the following set of constraints:

1. One timestep
2. One cell
3. Legal moves

This is implemented in

3 Efficient solution

While the naive solution works well in general, for boards ranging up to ... x ... it has a quadratic number of constraints for encoding the one cell and one time only. A more efficient solution would benefit of having less constraints for reducing the search space of the solver. After soing some research, I gathered information about many encodings [1] for the "at-most-one" family of constraints, refined to "exactly one" encoding in the case of this problem. It appears my initial encoding is actually known as "pairwise encoding". The sequential counter seemed to be the most effective encoding. The sequential counter works by assigning a set of new, auxiliary variables, whose size grows *linearly* with the size of the problem.

This technique can be applied to both the one cell only and one timestep only constraints.

However, it appears that the sequential counter encoding is not always better than the pairwise encoding, as reported by [1] Indeed, because of ...

This is implemented in

4 Counting solutions

In order to count all solutions for a given board, one simply needs to enumerate all the models found by the SAT solver, for each couple $(i_0, j_0) \in M \times N$. Using the PySAT library, it is easy to do so by calling the `enum_models()` method after solving the problem and simply count them.

Comparing the results obtained when solving 3×4 as well as 5 chessboards with the ones showed in [2]

This is implemented in

5 Counting up to symmetry

The easiest way to count up to symmetry was to enumerate all the solutions, apply symmetries and check whether or not the solution is identical to another solution.

This is implemented in

6 Finding additional constraints

References

[1] Holger H. Hoos and Thomas Stützle. Sat solvers for the knight's tour problem. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, pages 723–731. Springer, 2007.

[2] Wikipedia contributors. Problème du cavalier. https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_cavalier#D%C3%A9nombrement_des_solutions, 2025. Accessed: 2025-11-13.

[3] Wislez Valérian. knight_tour: Project for sat solving the knight's tour problem. https://github.com/valerian20024/knight_tour_SAT, 2025. GitHub repository, Accessed: 2025-11-13.