instituto superior de engenharia de lisboa

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores
Licenciatura em Engenharia Informática e de Computadores
Licenciatura em Matemática Aplicada à Tecnologia e à Empresa
Licenciatura em Engenharia Informática, Redes e Telecomunicações

Second Practical Project

Artificial Intelligence course

2024/2025 Summer Semester

Version 1.00

Teacher: Nuno Leite

ISEL, 1 May 2025

instituto superior de engenharia de lisboa

Artificial Intelligence course
Second Practical Project
2024/2025 Summer Semester

Learning objectives

At the end of the **Second practical project**, students should be able to:

- ☐ Approach Problem-Solving as Search, using basic (non-informed) and more advanced algorithms (informed);
- ☐ Understand the concept of *state space*;
- ☐ Apply non-informed and informed search algorithms to solve a Sokoban-based game;
- ☐ Approach Problem-Solving as search in a solution space;
- ☐ Understand optimisation algorithms that explore a space of solutions, such as *simulated annealing* and *genetic algorithms*;
- ☐ Apply *simulated annealing* and *genetic algorithms* to solve a Sokoban-based game.

Kurtan game

In this project, your group will program an *automated solver* for the famous *Kurtan* game. The Kurtan game is a variation of the Sokoban game. The word Sokoban literally means "depositor warehouse". Figure 1 illustrates a given phase of the game.

The objective of the game is to move the character (in the directions North, South, East and West) in order to push the wooden boxes to one of the final positions (green dots in the figure). The room is bordered by walls and corridors. The character can only push a box (not pull it). In the room there are gates that have to be opened with a key. The respective keys appear in a given position of the room automatically as the character arranges a certain number of boxes in the final positions. Once the player picks up a key, one of the gates opens. The question mark represents the final position where the character has to go once he/she has arranged the boxes. The stairs and paths lead to other levels, which should be ignored when solving the exercise (consider that there are walls at the end of stairs or paths). Of course, if you want, you can implement these aspects and game levels. More information about the Sokoban game can be consulted in: `https://en.wikipedia.org/wiki/Sokoban`. You can play the Kurtan game at: `https://www.myabandonware.com/game/kurtan-1vs/play-1vs`.

Your assignment is to develop programs (in Prolog and in Octave/Matlab) that demonstrate the behaviour of different solvers, namely solvers that employ the following algorithms:

- iterative-deepening (non-informed), programmed in Prolog;

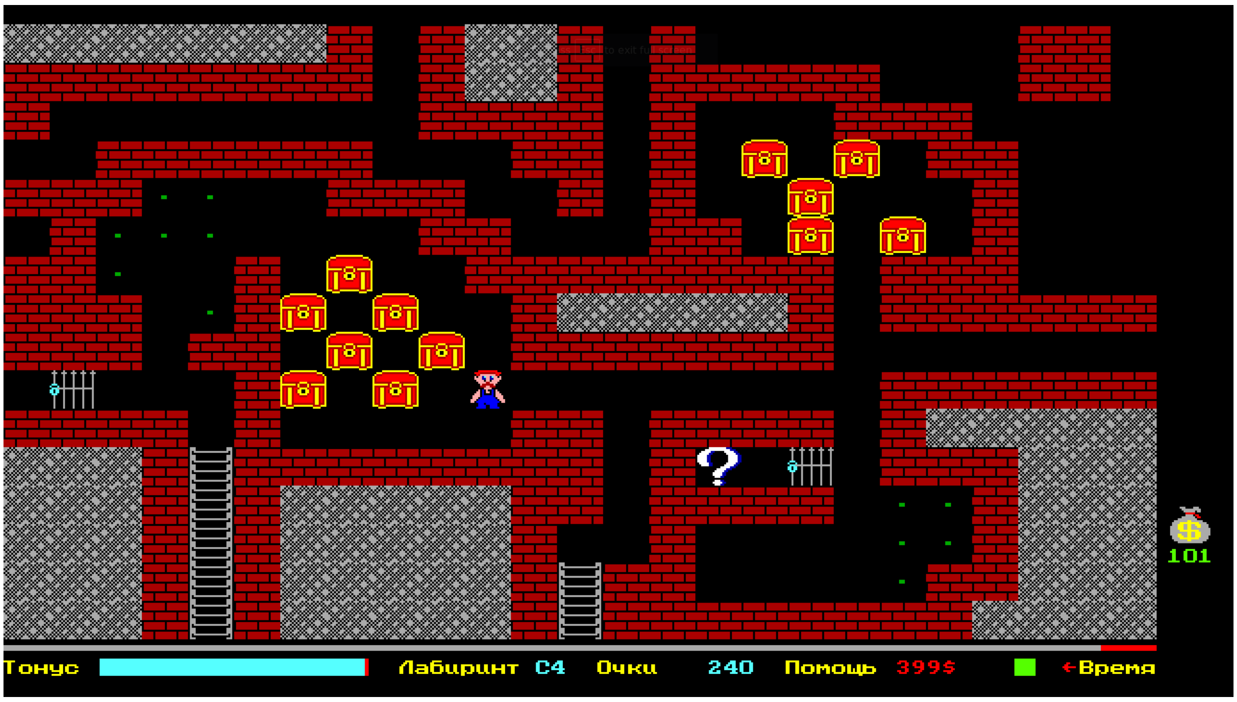

Figure 1: Illustration of a level in the game Kurtan.

- best-first search or A* (informed), programmed in Prolog. You have to devise a proper *admissible heuristic*;
- simulated annealing (optimisation algorithm);
- genetic algorithms (optimisation algorithm).

These algorithms will be available in the GitHub's Artificial Intelligence course site, and are written in Prolog and Octave/Matlab languages.

The input/output is text-based.

You should demonstrate the behaviour of the automated solvers for some example levels, starting with a very small level with just one box, and then increasing the difficulty and number of boxes. You don't need to implement level loaders (the levels could be hard-coded for simplicity), but if you want to do so, you could use the level format published in: `http://www.sokobano.de/wiki/index.php?title=Level_format`.

In `https://en.wikipedia.org/wiki/Sokoban`, you can find some example Sokoban levels that you can adapt.

NOTE 1: It is suggested that you implement a logic that dynamically changes (through the `assert` and `retract` predicates), the map that you define. This logic will add key information and removal of gates from the map.

NOTE 2: Due to the implementation details of *simulated annealing* (*SA*) and *genetic algorithms* (*GA*), it is suggested that you use the available Octave/Matlab code and implement the operators using this language also.

NOTE 3: Despite the player cannot pull boxes, the SA and GA move operators can generate situations where a given box is pulled.

Due date: 31 May 2025 until 23:59.

The delivery of the work must present the report, Prolog code (solver using search algorithms), and other developed code (solver using optimization algorithms, *simulated annealing* and *genetic algorithms*), delivered in the Moodle system. The report must be concise and justify all decisions taken. It must indicate the student group composition and the curricular unit info.