CS4402 - Practical 1: Bombastic

170008773

13th February 2018

Abstract

Keywords:

1 Introduction

2 Problem description

2.1 The game

Bombastic is played on a $N \times M$ grid of cells. The cells in this grid can either be `dead`, `ice`, or `normal`. `dead` cells cannot be entered by anything, and other cells can accomodate at most one block or the avatar. For simplicities sake we assume that every grid is surrounded by a wall of dead cells. On this grid there is an avatar and one or more blocks and a number of goals, `equal to the number of blocks`. The objective of the avatar is to walk around the gird and push the blocks around until all blocks are at a goal. In this scenario we are not interested in which block ends up at which goals.

2.2 Avatar logic

The avatar is allowed to move around the grid, moving the cell it is currently occupying to any of the adjacent cells that are not dead. The avatar is only allowed to move purely horizontally or vertically (i.e. not at the time) and not moving is also disallowed. The player is allowed to move onto an ice cell, however when it moves off that cell again, the ice will crack, turning the cell into a dead one.

2.3 Box logic

The avatar can move blocks by moving into their square. This will move the block one square in the direction the avatar is travelling in. This is only allowed if the cell the block is moving into is not dead and does not contain another block.

2.4 Objectives

Given the grid layout, the positions of the blocks, the position of the goals, and the initial position of the avatar, the objective is to find a sequence of legal moves that move all the blocks to a goal. In this instance the number of moves is provided. So the problem is to find whether there is a sequence of the given length that satisfies all the objectives.

3 Modeling the problem

3.1 Setup

In this instance I was required to use the modelling language `Essence prime` in conjunction with the constraint solver Savile Row. I was also provided with the decision variables, their domains and several sets of parameters to test the system.

3.2 Designed instances

I was also required to design new instances of this problem class. I took this opportunity to design a few instances that can be used to either test specific parts of the modelled logic or some artificially difficult problem to test the performance. They are designed in pairs. Every problem has one paramter file that is solvable and one that isn't so that we can see how the performance compares to comperable problems. We will discuss them below

2Blocks

BigComplex

iceT

L

T

WideOpen

4 Emperical evluation

5 Conclusion

word count: