COMP4075/G54RFP Coursework Part III

Benjamin Charlton — psybc3 — 4262648

16th January 2019

1 Project Overview

1.1 Motivation

The original basis for this project comes from a series of lab exercises from the G52AIM module, Artificial Intelligence Methods. The lab exercises involved implementing a variety of AI methods to solve some basic optimisation problems namely MAX-SAT problems.

Many of the methods implemented involved combining smaller functions together to create the desired effect. This could effectively translated into a functional programming setting. I thought it would be interesting to try and reimplement these some of these methods in Haskell to see the benefits of the FP paradigm to these AI methods.

1.2 Technical Background

1.2.1 MAX-SAT

MAX-SAT is an optimisation problem which is NP-Hard. Given a logic formula in conjunctive normal form, the aim is to maximise the number of clauses which are true after variable assignment. A logical problem in conjunctive normal form has the following formal grammar.

```
CNF -> (C)  |  (C) & CNF        L -> V  |  !V
C   -> L    |  L + C            V -> 1  |  2  |  3  |  ...
```

CNF Formula A CNF formula is made up of clauses which are logically and'd together. If each clause in a CNF problem is true then the whole formula will return true. Although in MAX-SAT we aren't aiming for all clauses to be true only to maximise the number that are true.

Clauses C Each clause contains a series of literals, which are logically or'd together. This means so long as any literal can evaluate to true the whole clause will be true.

Literals L A literal is a variable which is either positive or negative, this is achieved by applying a logical not or not doing so.

Variables V Finally each literal is assigned a variable which are later used to evaluate a solution. There can be any number of variables so integers are used here in this example to represent the different variables.

Solution for CNF A solution for CNF will involve all of the variables in the problem being given a boolean value. From here you can evaluate each clause and then find how many clauses evaluate to true. One benefit of how this problem works is that any solution (granting that all variables required have an assignment) will be a valid solution to the problem.

1.2.2 Mathematical Optimisation

The MAX-SAT problem is a form of optimisation problem that can be solved by a optimisation technique. An optimisation problem means that there is an objective function that you wish to maximise the value of by changing the input. In the case of MAX-SAT the objective function is the number of clauses that evaluate to true given our solution as input. The goal of any optimisation problem is to find the solution that gives the best possible value for the objective function.

In MAX-SAT there may be several optimal solutions and it is hard to calculate if you do have an optimal solution. Apart from the trivial case of all clauses being true there is no guarantee that an optimal solution has been found, there is also no guarantee that all clauses can be true either.

While you could devise a way to generate all possible solutions and evaluate them all to find the optimal, due to the NP-hardness of this problem this method will take none polynomial time to run which is very inefficient. The field of mathematical optimisation aims to find techniques to more efficiently find optimal or near optimal solutions

to these types of problems. While the techniques used for this project don't guarantee optimal solutions the achieve near optimal solutions that have been shown to produce solutions very close to the true optimum. Although MAX-SAT doesn't have any immediately obvious applications it is very similar to problems such as scheduling where near optimal solutions may be sufficient so the application of these algorithms have real world importance.

1.3 Aims of the Project

The original coursework took place over several lab sessions, incorporating a variety of topics and theoretical questions, as well as originally relying heavily of a java based framework. Due to this only part of the lab exercises will be looked into and some parts of the java framework will have to be remade in Haskell.

Here is what I intend to create in this project:

- MAX-SAT problem generator
- MAX-SAT evaluator
- Naive solvers
- Hill Climbing solvers
- Genetic Algorithm Solver

While creating this components, real world functional programming techniques will be applied to make this work efficient and maintainable.

2 Implementaion

In this section the implementation of each component will be discussed including any key decisions and interesting aspects.

2.1 MAX-SAT problem generator

In the java framework there was the ability to generate a MAX-SAT problem in a suitable form, before any solvers could exist this component and decisions of how the problem will appear also had to be made.

2.2 MAX-SAT evaluator

2.3 Naive solvers

2.4 Hill Climbing solvers

2.5 Genetic Algorithm Solver

The MAX-SAT problems were generated by a framework given in the original courseworks, this was via a java file that could be imported.

3 Learnt stuff — Needs another name

A section reflecting upon what was learned from the project and your thoughts around the project topic from a real-world programming perspective.