Resolution of a Decision Problem using Logic Programming with Restrictions: Crypto Equations

Pedro Ferreira (up201806506) and Ricardo Amaral Nunes (up201706860)
FEUP-PLOG, 3MIEIC03, Group crypto-product_4

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, 4200-465 PORTO

Abstract. The project consists in a program, written in PROLOG, that can solve a multiplication equation in the form of letters. Every letter in the problem consists in a different number, however digits with the same colour correspond to the same one. From the several search strategies tested, the one that yield better results was the first-fail in conjunction with bisect.

Keywords: Crypto Equations · sicstus · prolog · feup.

1 Introduction

In this project, a multiplication problem will be addressed and the solutions found using prolog with restrictions will be studied and tested.

This multiplication problem consists in matching coloured circles with digits, making the given equation true. This equation is composed of three different numbers, being them the two multiplied numbers and the result of that operation. All circles of the same colour have to correspond to the same digit and each colour must represent a different one.

This kind of problem is referred to as Alphametic (also called cryptarithm) Multiplication Questions.

This article has the following structure:

- Problem Description: Describes in detail the problem of optimization or decision under analysis, including all restrictions involved.
- Approach: Describe the problem modelling as a PSR:
  - Decision Variables: Describes the decision variables and respective domains, as well as their meaning in the context of the problem under analysis.
  - Constraints: Describe the problem's rigid and flexible constraints and the implementation using the SICStus Prolog.
- Solution Presentation: Explain the predicates that allow inspecting the solution in text mode.
- Experiments and Results:

  - Dimensional Analysis: Includes examples of execution in instances of the problem with different dimensions and analyzes the results obtained.
  - Research Strategies: Different Research Strategies (heuristics of choice of variable and value) used and comparison of results obtained
- Conclusions and Future Work: Conclusions drawn from the project, results obtained, advantages and limitations of the proposed solution, aspects to be improved

2 Problem Description

The problem approached in this paper is called Crypto-product. This problem presents a multiplication equation of any length, with exactly two parcels and with the digits substituted by coloured circles. The goal is to substitute each circle a digit in order to make the equation valid. The following image illustrates an example of the problem.

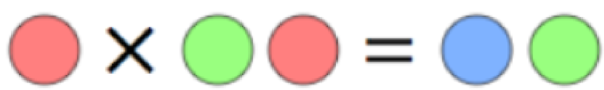

Fig. 1. Example of crypto-product problem [1]

There are some restraints to the problem, both explicit and implicit. To begin with, all circles with the same colour must represent the same digit. In addition, two circles of different colours must represent different digits. A more implicit restraint is the first digit of each parcel must be different than 0. Finally, the multiplication must be valid. In the example above, there are a total of five digits but only three unique: one represented by the red circle, other by the green one and the third one represented by the blue. Moreover, the red and green digit[1] must be different than 0.

3 Approach

3.1 Decision Variables

In this problem, the decision variables are the digits that each colour represents. To facilitate their manipulation, they are divided into three lists, one for the expression at the left of the multiplication, one for the right one and another for the solution. There are no limitations in the number of variables.

Since the decision variables are digits, they all have the same domain, which is 0 to 9.

[1] To simplify, for the rest of the paper, a red, green or any other color digit corresponds to the digit that the circle with that color represents

3.2 Constraints

The first restraint applied is that different colour circles have different digits, with the predicate all_distinct/1. Next, it is guaranteed that the first digit of each expression is different from 0, by using the predicate first_diff_zero(+List), that place the restraint #=0 to the first element of the list. After this, all three expressions are converted to numbers, using the exp_to_number(+Expr, -NumRes) predicate. It converts the expression (a list) consisting of unbound variables into a number, multiplying each variable by 10 to the power of its position in the list and summing them all. All the operations of this predicate use the equality restraint #=. Finally, after converting the expressions into numbers, the restraint that guarantees the validity of the equation is placed, just by multiplying the numbers and applying the equality restraint with the expected result.

4 Solution Presentation

When starting the program, the initial menu appears. It consists of the name "Crypto-number" in ASCII art and the list of available problems. This is made possible by the menu/0 predicate. To view the solution to the problem, it is necessary to select the problem by the number.

Fig. 2. Initial Menu

The problem will then solve the problem in run-time and present the solution in the following format:

```
The equation chosen was: G * GB = BBR
The corresponding solution is: 9 X 98 = 882
```

Fig. 3. Solution to problem 1

By selecting the 'Generate Puzzle' option and introducing the size of the pretended puzzle, it is possible to create one at run-time.

```
Write the size of the puzzle: |: 5.
Orange * PinkOrangeGreenRed = YellowBluePinkGreenOrange
Press anything to show the solution|: y.
The corresponding solution is: 2 X 5296 = 10592
```

Fig. 4. Generate puzzle menu

5 Experiments and Results

5.1 Dimensional analysis

In a problem of this nature, the amount of distinct aspects to take into consideration is very high, being difficult to find the correct equations to compare and conclude precisely upon the subject. To better understand the influence that the number of colours brings to the problem we started by gathering the results shown in Fig 5, Table 2 (Hyphen cells mean the number of colours would be higher than digits, which is impossible) and Table 6.

In every test, the time elapsed, number of total backtracks and the number of restrictions was measured and analysed. For these tests, the times were recorded using the default labelling flags, which means, using the leftmost option when it comes to sorting and step as value selection. Each test was performed three times, and the result is the average of each try.

To determine the factors that influence the time by the solver, we increased the number of different letters in each equation, maintaining the same number of total digits. By doing so, we concluded that the number of colours does not influence in a very significant way the efficiency of the solution when dealing with small equations, being the calculation almost instantaneous.

In the other hand, when dealing with larger equation, the time elapsed increases substantially. By examining the values more carefully, we concluded that the number of digits with the same colour in the equation decreases the total time taken to find a solution. It can also be noted that when the number of colours is close or equal to the total number of digits, the speed of the execution decreases.

In conclusion, the number of colours in the equation has minimal influence over the duration time. The main factor is indeed the length of the equation.

5.2 Search Strategies

Since the argument Options from the predicate labeling/2 controls the variables and values order of selection, we tried different approaches and combinations to increase the efficiency of the program.

Thusly, we started by changing the sorting variable and gathered results using the strategy ff, instead of leftmost. It is to note that, the difference between this two resides in the fact that the second one, not only chooses the leftmost variable, but also chooses it only from the ones with the smallest domain (Table 3, Fig 6).

With this test, we could clearly see an improvement in the efficiency of the program. After further study, we determined the numbers of backtracks in this second experience is lower than the numbers previously seen. These results are achieved due to the decrease of the number of tests to be done, since we discard the solutions with higher domains. (Table 7)

We now tried to change the strategy of value selection. Hence, the following table represents the values gathered when using the median/middle strategy combined with the sorting strategy ff, that was previously confirmed as more appropriate than the others tested and henceforward will be used in all the tests to come (Table 4).

Although for more trivial equations this strategy is almost identical to the one used previously, it is clearly not ideal for our problem. When analysing the number of backtracks needed to reach the solution, we can see it increases considerably compared to other strategies. (Table 8)

After this failure, we tried a more open strategy to solve the efficiency problem and went for the bisect value selection option. In this strategy, as seen in Theoretical classes, for each variable X, a choice is made between X #=¡ M and X #¿ M, where M is the midpoint of the domain of X (Table 5, Fig 7).

The efficiency difference between this strategy and the others is notorious. While the previous one's times are in the order of seconds, the last one makes every calculation used in these tests almost immediate, will all the times relatively the same.

Furthermore, by checking the number of backtracks needed to reach the solution, we can see this number decreased more than 1000%, being the number of restrictions, exactly, the same. These results confirm our suspicion that the main efficiency factor is the number of backtracks and not the restrictions themselves. (Table 9)

6 Conclusions and Future Work

With the elaboration of this project and the application of theoretical and practical classes knowledge, it was clear to us that restrictions are a very useful tool when dealing with decision and optimization problems. Its combination with Prolog also proved to be very time efficient.

By the tests and results obtained, we concluded the limiting factor for the program's efficiency is the number of backtrackings and not the number of restrictions as one could be led to think at the beginning.

Overall, the solution found is very efficient and solves the equations very quickly. However, the main limitation we can find is that for equations with big numbers it will depend on the memory capacity of SICStus.

In short, the project was successfully concluded, as a solution for the initial problem was found. Furthermore, it helped us have a better understanding about the way labelling and decision variables function.

References

1. Crypto Equations rules, https://erich-friedman.github.io/puzzle/crypto/
2. SICStus Prolog User's Manual (Release 4.6), Section 10.10: Constraint Logic Programming over Finite Domains—library(clpfd)
3. SICStus Prolog, https://sicstus.sics.se/

7 Annex

Table 1. Example tests used to obtain results

	6 Colours	8 Colours	10 Colours
10 Digits	0.03s	0.02s	0.06s
12 Digits	0.06s	0.06s	0.04s
14 Digits	0.07s	0.07s	0.08s

Table 2. Elapsed time for the number of different colours and total digits in the equation, using default labeling/2 options

	3 Colours	4 Colours	5 Colours	6 Colours	7 Colours
5 Digits	0.06s	0.06s	0.06s	-	-
6 Digits	0.04s	0.07s	0.07s	0.06s	-
7 Digits	0.06s	0.06s	0.04s	0.05s	0.07s

	6 Colours	8 Colours	10 Colours
10 Digits	0.12s	0.11s	0.13s
12 Digits	0.29s	1.87s	1.34s
14 Digits	2.46s	7.47s	4.17s

Table 3. Elapsed time for the number of different colours and total digits in the equation, using ff labeling/2 option.

	6 Colours	8 Colours	10 Colours
10 Digits	0.06s	0.09s	0.10s
12 Digits	0.08s	0.07s	0.07s
14 Digits	0.15s	1.94s	0.97s

Table 4. Elapsed time for the number of different colours and total digits in the equation, using ff and median/middle labeling/2 options

	6 Colours	8 Colours	10 Colours
10 Digits	0.06s	0.09s	0.15s
12 Digits	0.11s	0.08s	2.36s
14 Digits	0.28s	18.05s	5.71s

Table 5. Elapsed time for the number of different colours and total digits in the equation, using ff and bisect labeling/2 options

	6 Colours	8 Colours	10 Colours
10 Digits	0.03s	0.02s	0.06s
12 Digits	0.06s	0.06s	0.04s
14 Digits	0.07s	0.07s	0.08s

Table 6. Number of backtrackings for the number of different colours and total digits in the equation, using default labeling/2 options

	6 Colours	8 Colours	10 Colours
10 Digits	490	5890	5695
12 Digits	71391	160756	80876
14 Digits	403746	1378976	707986

Table 7. Number of backtrackings for the number of different colours and total digits in the equation, using ff labeling/2 option.

	6 Colours	8 Colours	10 Colours
10 Digits	1566	1933	4742
12 Digits	303	565	438
14 Digits	12723	245309	175793

Table 8. Number of backtrackings for the number of different colours and total digits in the equation, using ff and median/middle labeling/2 options

	6 Colours	8 Colours	10 Colours
10 Digits	10958	1634	8031
12 Digits	17128	951	452045
14 Digits	41522	2758659	1261569

Table 9. Number of backtrackings for the number of different colours and total digits in the equation, using ff and bisect labeling/2 options

	6 Colours	8 Colours	10 Colours
10 Digits	0	13	71
12 Digits	19	57	1
14 Digits	108	19	135

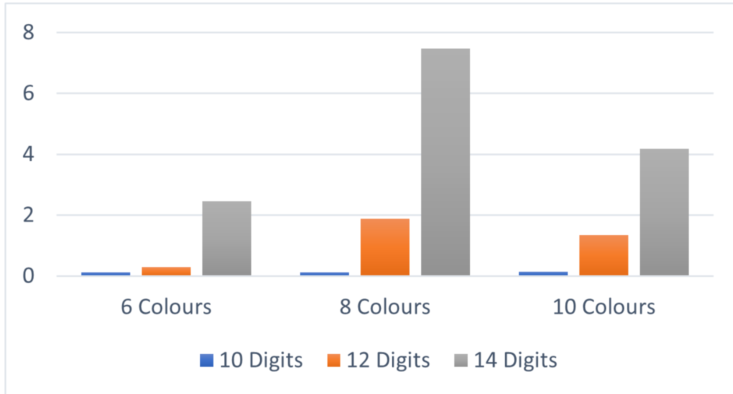

Fig. 5. Graphical representation of the elapsed time correspondent to Table 2.

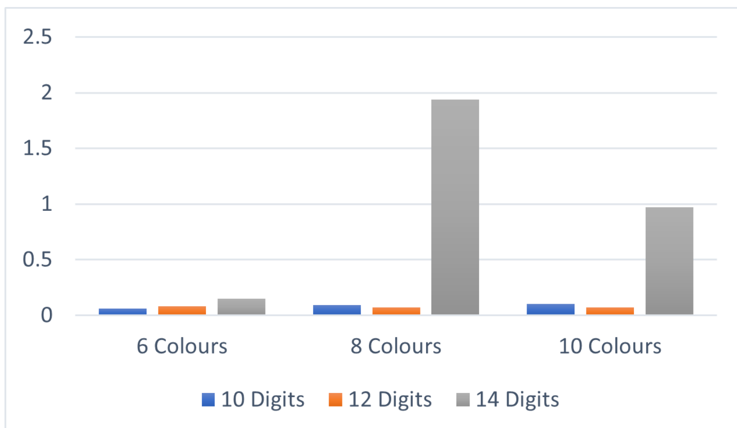

Fig. 6. Graphical representation of the elapsed time correspondent to Table 3.

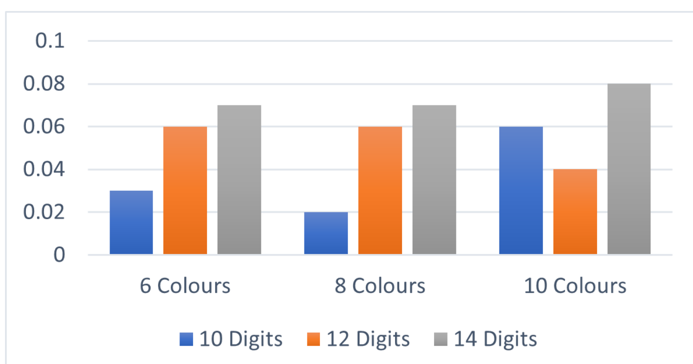

Fig. 7. Graphical representation of the elapsed time correspondent to Table 5.