UNIVERSITAT POLITÈCNICA DE CATALUNYA

DELIVERABLE 1: CONTEXT AND SCOPE OF THE PROJECT

Design of an environment for solving pseudo-boolean optimization problems

Author:
Marc BENEDÍ

Supervisor:
Dr. Jordi CORTADELLA

GEP

March 6, 2018
Edinburgh, UK

Contents

Chapter 1

Introduction

1.1 Context

Before explaining the main problem which this project is about, *Pseudo-Boolean Minimization*, it necessary to do a quick introduction into a much wider topic.

Boolean satisfiability problems *(SAT from now on)* is the problem of finding a model[1] for a *Boolean Formula* (BF from now on). In other words, it is the result of evaluating the *BF* after replacing its variables for *true* or *false*.
SAT is widely used in Computer Science because it was the first problem proved to be NP-Complete[1][2] which allowed a lot of NP[3] problems be reduced to it.

1.1.1 What is a Pseudo-Boolean Formula?

In propositional logic, a *BF* is defined as following[2]:
Let P be a set of predicate symbols like $p, q, r, ...$

- All predicate symbol of P is a formula.
- If F and G are formulae, then $(F \wedge G)$ and $(F \vee G)$ are formulae to.
- If F is a formula, then $(\neg F)$ is a formula.
- Nothing else is a formula.

This representation has some limitations because it can only express properties which are *true* or *false*.

Pseudo-Boolean Formulas are functions of the form $f : B^n \rightarrow \mathbb{R}$. For example, the following formula is a *Pseudo-Boolean Formula* (PBF from now on): $3x + 5y$. Therefore, *BF* are a special case of *PBF* where the domain is $d = \{0, 1\}$.

1.1.2 Pseudo-Boolean formulae minimization

PBF minimization is a well known NP-Hard[4] problem.
It does the following:
Given a *PBF* of the form $\sum_{i=1}^{n} x_i w_i \leq k$, where $w_i, k \in \mathbb{I}$ and $x_i \in \{0, 1\}$, it tries to find

[1]An interpretation which satisfies the formula.
[2]NP and NP-Hard.
[3]Nondeterministic polynomial time.
[4]NP-Hard: at least as hard as the hardest problems in NP (more)

the minimum *k* which satisfies the constraint.

There is a big research on this field, more specifically in encoding *PBF* into *CNF*. In this paper, Hölldobler, Manthey, Steinke[3], some relevant *PBF* into *SAT* encodings are explained and a new one is proposed. One of the authors of this paper, Steinke, is also the author of *PBLib*.

1.2 Background

During the past semester (Q1 2017/2018), under the supervision of Dr. Jordi Cortadella, I had been developing a C++ library.
This tool allows the users to represent *BF* in a C++ program in a intuitive way, do operations between them and convert them into *Binary Decision Diagrams* (BDD from now on). However, the main functionality of this library is the conversion from a *BF* to *CNF*.
As previously explained, *CNF* is a particular type of a *BF*, a conjunction of disjunctions. *CNF* is an important format because it is the standard input for *SAT Solvers*A.1.
As shown in this paper, *Mitchell, Selman, and Levesque[4]*, there is a correlation between the number of variables, the number of clauses and the hardness of solving the *CNF*.

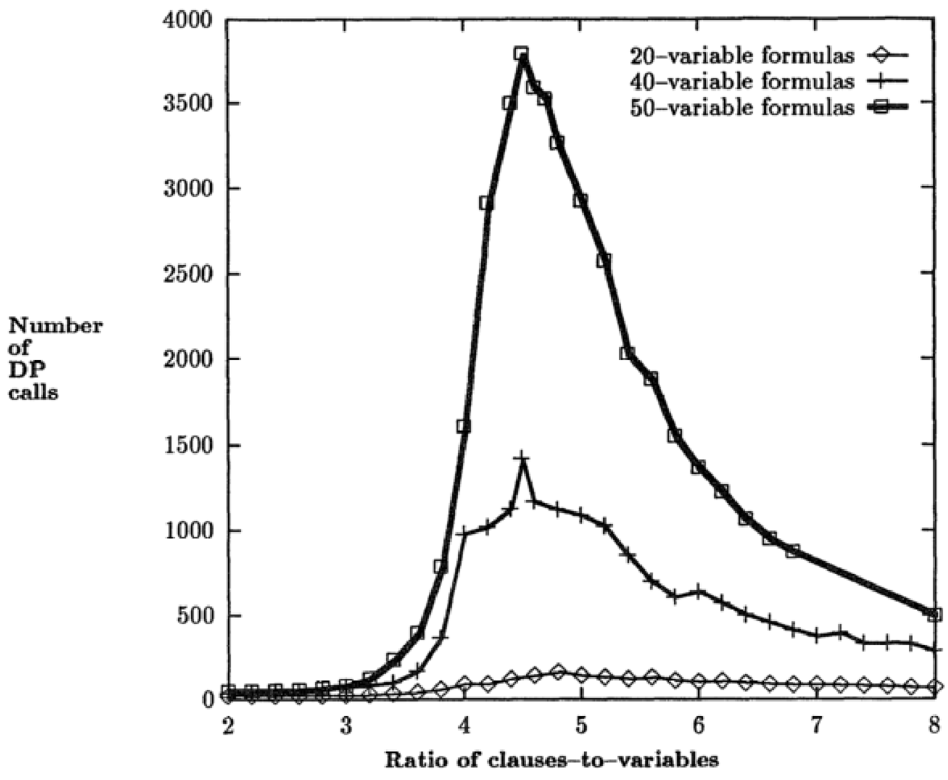
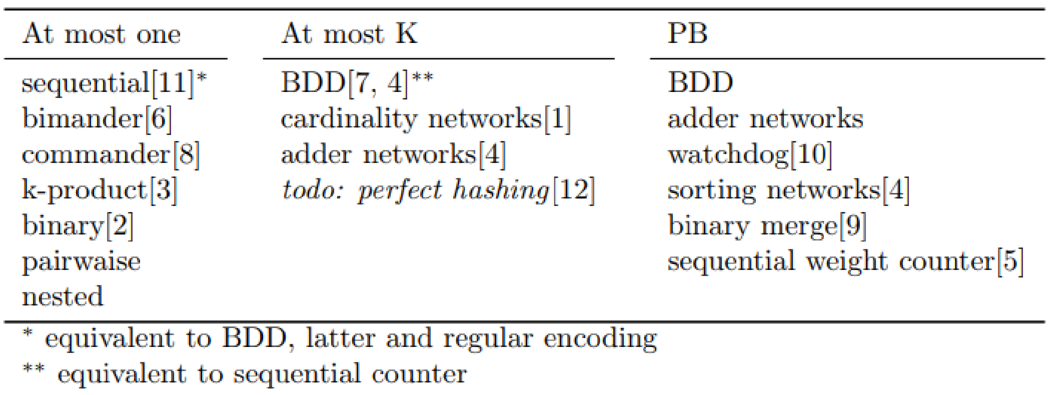


FIGURE 1.1: Median number of recursive DP calls for Random 3-SAT formulas, as a function of the ratio of clauses-to-variables. Extracted from Mitchell, Selman, and Levesque[4]

Therefore, an improvement of the input *CNF* of the *SAT Solver* can reduce a lot the hardness of the problem.
This is the main goal of the library, try to reduce the size of the final *CNF* resulting from applying different converting methods on the original *BF*.

1.3 Sate-of-art

In this section previous projects are discussed.
The first one is PBLib. PBLib is a C++ toolkit for encoding *PB Constraints* into *CNF*. As explained in *Steinke*[5], PBLib implements a lot of encodings:



At most one	At most K	PB
sequential[11]*	BDD[7, 4]**	BDD
bimander[6]	cardinality networks[1]	adder networks
commander[8]	adder networks[4]	watchdog[10]
k-product[3]	*todo: perfect hashing*[12]	sorting networks[4]
binary[2]		binary merge[9]
pairwaise		sequential weight counter[5]
nested		

* equivalent to BDD, latter and regular encoding
** equivalent to sequential counter

FIGURE 1.2: PBLib implemented encodings
Extracted from Steinke[5]

PBLib does not only implement this encodings, the most interesting thing is that it can decides which encoder provides the most effective translation.
It is very hard to compete on this aspect against PBLib, but it is not very user friendly. For this reason, on of the goals of this project is to add a layer between the user and PBLib to simplify how the user declares the *PB Constraints*.

The second project we will talk about is the explained in *Background section*. This project adds a new encoding for *BF* which can be extended to *PB Constraints*. This encoding will be studied and if it achieves good metrics it will be implemented to *PB Constraints*

1.4 Motivation

Informatics Logic is taught in this[5] faculty. In that course I realized how important is *logic* through its lecturer, Dr. Robert Nieuwenhuis, and its activities.

In the first coursework we had to code a *SAT Solver* which used *Unit Propagation*. With this activity I comprehended how hard and substantial is the study of *logic* and all its context. For example, how *logic* is used in Artificial Intelligence and Planners.

When the time of deciding the *TFG* arrived , I contacted my actual supervisor, Dr. Jordi Cortadella, and he proposed me some topics and ideas. Finally, we agreed on doing this project.

The motivation for this project is try to deepen into the topic and contribute on it.

[5]Facultat Informàtica de Barcelona

1.5 Stakeholders

In this section the Stakeholders of the project are defined. Stakeholders are entities which are effected, directly or indirectly, by the solution developed in this project.

1.5.1 Target audience

This tools tools targets all the entities (researchers, companies, ...) which works with *PB minimization* and use *SAT Solvers*.

1.5.2 Users

The users will be C++ programmers due this tool is developed in this language.

1.5.3 Beneficiaries

All those entities which works with *PB minimization*. For example AI, SAT Solvers, Planners, ...

Chapter 2

Project Formulation

As mentioned before1, this project is an extension of a previous C++ library. The main goal of this project is improve the time required to solve a *minimization* problems. To achieve this goal, the following objectives have been established.

2.1 General objectives

2.1.1 Pseudo-Boolean minimization

For the problems of the form $min(c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq k)$, the goal is to find an assignment for $\{x_1, x_2, \ldots, x_n\}$ so that k is minimum.

Previously1, it has been explained that this types of problems are *NP-Hard*. This project will try to reduce the time to solve this problems through two approaches:

- Binary search:
  Implement the well known *Binary Search*[1] algorithm to find the minimum value for k.

- Linear search:
  Some *SAT Solvers* can learn and derive new restrictions from previous problems. To take advantage of this ability it is necessary to implement a *Linear Search* algorithm.

2.1.2 Timeout

For some problems it is more important to find a solution before a deadline than finding the best possible solution. For instance, a delivery company must have all the route planned for all trucks before the journey starts, therefore, they care more about having a solution than finding the best one.
For this, a *Timeout strategy* will be implemented in case that a good enough solution has been found or the problem does not seem to have one.

2.1.3 Multi-threading

This tool will take advantage of multi-core processors trying to split the problem and solving each part separately.

[1]Binary search is a search algorithm that finds the position of a target value within a sorted array. (more)

Chapter 3

Scope

3.1 What and how?

To achieve all the general objectives2 of the project, the following stages have been established:

- Analyze, refactor[1] and test the existing code to have a solid base.
- Add the functionality of representing *PBF*.
- Study PBLib library to see which functionalities it has available to work with *minimization*.
- Implement *minimization* strategies.
- Study timeout strategies and implement them.
- Study and implement multithreading.

3.2 Possible obstacles

In this section the possible obstacles and its solutions are exposed.

3.2.1 Base project

This project will be built on top of an existing one, as explained in *Background section*1. The existing project could be a source of bugs and other problems caused by not following an adequate methodology. For this reason and to solve possible issues, the first stage of the project will be focused on solving them.

3.2.2 Schedule

Due to the circumstances in which this project will be developed (Erasmus) possible delays could appear. To fix this circumstances, a realistic schedule with weekly communication will be planned. This will support a continuous development and detect as soon as possible delays.

3.2.3 PBLib

One of the main requirements of this project, *Pseudo-Boolean minimization*, is planned to be done with *PBLib* library. It may be this library does not fit as expected with the project forcing to find a substitute.

[1]Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior. (more)

3.2.4 Correctness

As explained in *Rigor and Validation*4, correctness in this project is very important because of the context it is in.
Guarantee correctness could be hard and take more time than expected. If this happens, formal correctness could be delayed or reduced.

Chapter 4

Methodology and Rigor

Research is a vast process with no clear path between *a* and *b*. For this, it is important to follow some directions. Methodology will provide some guidelines to avoid possible problems, be more efficient and do the project more manageable.

4.1 Methodology

The methodology adopted for this project will be Agile[1]. It is important to clarify that this methodology will not be followed strictly but adapted to this particular case where there is only one developer and all the objectives are well defined. The main characteristics followed from Agile in this project will be:

- Short cycles
- TDD (Test-Driven Development)
- Weekly scrums with the supervisor

4.2 Tools

In this chapter the development tools will be introduced.

4.2.1 Git

Git will be used in this project as a Version Control System because it allows to maintain a tracking of all the changes made (commits), and what is more important, return to them at any time. In addition to this, it enforces a short cycle development (because commits are small units of work) and the developer has to document them which matches perfectly with Agile methodology. GitHub will be the repository service used.

4.2.2 Trello

Trello is a simple and flexible web board which helps to organize tasks and its state. It will be used in this project to manage tasks and priorities.

[1]Methodology based on the on the adaptability in front of any change to improve exit possibilities.

4.3 Communication

Due to my conditions, I'm currently studying abroad in an Erasmus program, all the communication will be made through electronic means. The majority of it will be made using e-mail but if it is necessary a video conference could be done.
The minimum communication with the supervisor will be a weekly e-mail report where all the tasks done during this period will be explained. Problems or questions will be also exposed, if any.

4.4 Rigor and Validation

Rigor and Validation for this project is relevant.
The surrounding of it, such as *Artificial Intelligence, Planners, Cryptographic Protocols verification, . . .*, are widely used nowadays and have been becoming more popular lately. This means that this project could have a big repercussion and be used by some professionals. For this, it is important to guarantee the validation and correctness of the project.
During the development, TDD will be used to avoid unnecessary code (possible origin of bugs) and assure the correctness of the implementation. It is also possible to formalize and prove all the operations done by the software.
Finally, my supervisor could give me orientation and validate, if necessary, the operations done.

Appendix A

More information

A.1 Why SAT Solvers use CNF as input format?

There are two mains reasons for this: Equisatisfiability and Computational Complexity. Let us start with the first one:

Two *BF* are **equisatisfiable** if and only if both have the same *models*. This may seem the same as equality but it is not because in an equality relationship both *BF* have to have the same variables.
This is important because between a *BF* and its result from a *CNF* transformation the equisatisfiability is preserved which means that if the *SAT Solver* finds a *model* for the *CNF*, then this *interpretation* will be also a *model* for the original *BF*.

The second reason is computational complexity. Let us have a look at the following table:

	DNF	CNF
TAUT	NP	P
SAT	P	NP

TABLE A.1: Complexity of deciding if a *BF* is SAT or TAUT depending of its format.

So as a *BF* can be converted into a *CNF* in linear time while preserving equisatisfiability, *SAT Solvers* will use them to target satisfiability.

Bibliography

[1] Stephen A. Cook. “The complexity of theorem-proving procedures”. In: *Proceedings of the third annual ACM symposium on Theory of computing - STOC ’71*. New York, New York, USA: ACM Press, 1971, pp. 151–158. DOI: 10.1145/800157.805047. URL: http://portal.acm.org/citation.cfm?doid=800157.805047.

[2] Rafael Farré et al. *Notas de Clase para IL - 2.Definición de la Lógica Proposicional*. Barcelona, 2009. URL: https://app.box.com/file/225148187559.

[3] Steffen Hölldobler, Norbert Manthey, and Peter Steinke. “A Compact Encoding of Pseudo-Boolean Constraints into SAT”. In: (). URL: https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2{\&}ik=7328377021{\&
}view=att{\&}th=15e2e7b559196d67{\&}attid=0.2{\&}disp=inline{\&
}safe=1{\&}zw{\&}saddbat=ANGjdJ9S{_}JppQCGQWibmAw70nm3SJsSOBhJg1PGGwc8D5mdov9JRPuNYNP
}A5wVWlyGHOKOsaPgNPEw3Y8EjoJoRyRKQO3MZA1Uma98pSK7xDtv5FPsTktHevmFB7ZP79m3vMbP7MvPgILPj
}2u{_}{_}eWwhfC14QE82TOnITUjuH7rGsdMKwsKwRrkwi-CoyPoJT8RAmSh2jKobXWTIejzVljGM8WWvOmg
}pgrV1LSjW4clhxqLfJfA-jW7Hi-AHADvop{_}Z{_}IiGErkOr-rhH9aMR7Z6-
pcC4aECx046{_}NOeCx5Fhmf-z71ZuWwUZ3PrMtEdDrDoJrmLDa2sWDQIuNom4yykZjVqBNSNoimvTOdUk31n
}CnkIjv5T15jmLv-hZGLO1f77keyiLsiHKGLZ{_}HAJGw9oLDXThmzABOivSmGjszgdj{_
}X1ODplVtdsqWtqViPDVmbctOtXfd33PYgBil3IO8hQ.

[4] David Mitchell, Bart Selman, and Hector Levesque. “Hard and Easy Distributions of SAT Problems”. In: (). URL: https://aaai.org/Papers/AAAI/1992/AAAI92-071.pdf.

[5] Peter Steinke. “PBLib – A C++ Toolkit for Encoding Pseudo-Boolean Constraints into CNF”. In: (2015). URL: http://tools.computational-logic.org/content/pblib/pblib.pdf.