SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics,

Verfication of selected NP-hard Problems

Zixuan Fan

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics,

Verfication of selected NP-hard Problems

Verifikation der ausgewählten NP-schweren Probleme

Author:	Zixuan Fan
Supervisor:	Tobias Nipkow PhD.
Advisor:	Katharina Kreuzer
Submission Date:	Submission date

I confirm that this bachelor's thesis in informatics, is my own work and I have documented all sources and material used.

Munich, Submission date Zixuan Fan

Acknowledgments

Abstract

NP-hardness is a widely discussed topic in the theoretical computer science. Since the publishing of the Cook-Levin-theorem in the early 1970s, the researchers have proved the NP-hardness of many problems. Among many NP-hard problems, the most famous are the 21 NP-complete problems given by Karp in the paper *Reducibility of Combinatorical Problems*, for it covers a considerable amount of NP-hard problems from various mathematical disciplines. The proofs of the NP-hardness, however, were limited on-paper. With the existence of the interactive theorem provers, it is possible to reproduce and verify the proofs with the aid of computers. In order to demonstrate the capability of interactive theorem provers in verifying the NP-hardness, we formalized and verified a few NP-hard problems using the well-known interactive theorem prover, Isabelle.

Contents

1. Introduction

1.1. Motivations

NP-Hard problems has been an fundamental problem in theoretical computer science since the 1970s, when Cook and Levin showed that the **Satisfiability** problem is **NP-Complete** and Karp gave a list of 21 **NP-Hard** problems. In the next few decades, many attempts were made to show that **P** = **NP** and to develop algorithms that computes **NP** problems efficiently. Among many fields related to **NP-Hardness**, we focus on the polynomial reductions, which show the **NP-Hardness** of decision problems.

All the existing proof of **NP-Hardness** were on-paper proofs, which lack the automated verification by a computer. With the existence of powerful interactive theorem provers, it is meaningful to formalize and verify the classical results of **NP-Hardness** in a computer, contributing to the theoretical basis of many existing formalisation results, e.g. cryptography, approximation algorithms etc. There has been an attempt to formalise **NP-Hard** problems that were given in Karp's paper in 1972. Our work benefits from this attempt and continues to formalise the rest of the 21 **NP-Hard** problems in the interactive theorem prover Isabelle.

1.2. Contributions

Our work contains 2 categories of problems.

1. Set Covering problems: **Exact Cover**, **Exact Hitting Set**
2. Weighted sum problems: **Subset Sum**, **Number Partition**, **Integer Programming**, **Knapsack**

For each listed problem, we present a polynomial reduction either from **Satisfiability** or from another problem that is listed above. Thus, a reduction trace from **Satisfiability** is witnessed. Furthermore, a proof for the soundness, completeness, and the polynomial complexity of each polynomial reduction is also presented.

1.3. Outline

In Chapter 2, we introduce the Isabelle dependencies and the work of the predecessors, on which our work is based. Furthermore, we present the mathematical background of the polynomial reductions and **NP-Hard** problems.

Chapter 3 and Chapter 4 follow with the formalisation and verification of the listed problems. For each decision problem, we start with the definition of the problem and the polynomial reduction. Then a proof of the soundness and completeness of the reduction is presented. To finish, a polynomial complexity of the reduction is also proved. In Chapter 3, we discuss the polynomial reduction of the set cover problems, while Chapter 4 consists of that of the weighted sum problems.

To finish, we conclude the current status of the project of formalisation of Karp's 21 **NP-Hard** problems and present a few possibilities for the verification of the rest of the problems in Chapter 5.

2. Preliminaries

2.1. Isabelle and Dependencies

Isabelle/HOL

Isabelle is a generic interactive theorem prover. HOL is the Isabelle's formalization of Higher-Order Logic, a logical system with inductive sets, types , well-founded recursion etc. Our implementation requires the introduction of new datatypes, formalisation of natural numbers and integers. Thus, this type system is necessary.

HOL-Real_Asymp and Laudau_Symbols

TODO

NREST

TODO

The Karp21 Project

The project aims to formalise all of the 21 **NP-Hard** problems in Karp's paper in 1972. Up till now, there are **TOCOUNT** problems of them finished, with a few other **NP-Hard** problems that are related but not in Karp's list. Our work also contributes to this project, formalising six of the remaining problems. Though dependent on this project, our work only reuses a few definitions by the predecessors, while the most formalisation and verification is original. An overview of the project is given in the following graph.

DigitsInBase

This entry of Archive of Formal Proofs shows the uniqueness of representation of natural numbers given an arbitrary base. In other words, it proves the well-definedness of the n-ary counting systems. Our implementation benefits from this repository in showing the correctness of the polynomial reduction from **Exact Cover** to **Number Partition**.

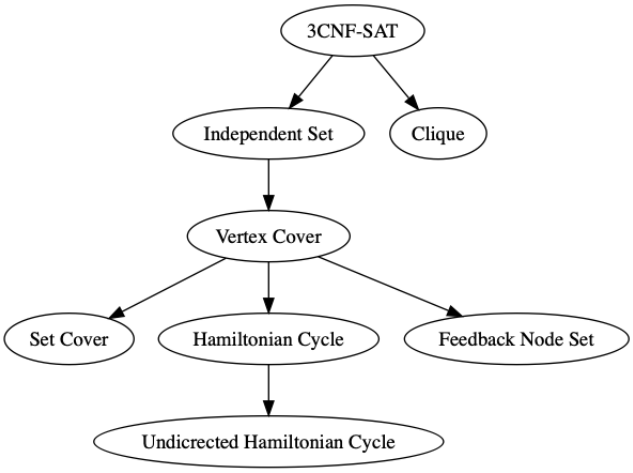

Figure 2.1.: Name

2.2. NP-Hardness and polynomial reductions

2.2.1. Decision problems

We may encounter many real-life problems that require a decision process to find a solution. When shopping at a supermarket, for example, we always want to choose the shortest queue. Another example is board games like go and chess. These problems are formally defined as decision problems.

A decision problem is a yes-no question on an infinite set of fixed type of inputs. Generally, if we refer to a decision problem A, we are referring to the set of all inputs for which the answer to the yes-no question is yes. The handling of a decision problem usually involves two questions:

1. Is there an algorithm, which computes the solution to this problem, terminating on all inputs?
2. If the answer to the first question is yes, is this algorithm efficient?

If the answer to the first question is yes for a problem, it is a decidable problem, otherwise it is non-decidable. We do not expect a yes or no answer for the second question, but would like to find the optimal complexity for the algorithm. If there is a non-deterministic algorithm that decides the solution to the problem in polynomial time, it is in the complexity class **NP-Hard**. Thus, we would like to know if the deterministic algorithms to compute the **NP-Hard** problems have a polynomial complexity.

2.2.2. Polynomial reductions

Given two decision problems A and B, a reduction is a function $f : A \to B$, which maps the inputs of the question of the first problem to that of the second problem. A reduction is

polynomial if and only if the reduction function has a polynomial bound. For a polynomial reduction from A to B, we writes $A \leq_p B$.

Let M and N denotes the domains of A and B respectively. A function $g : M \to N$ is a polynomial reduction if and only if the following conditions are fulfilled.

$$x \in A \iff g(x) \in B \tag{2.1}$$
$$\exists k \in \mathbb{N}. f \in \mathcal{O}(n^k) \tag{2.2}$$

For the convenience reason, we usually separate (2.1) into the soundness and completeness of the reduction.

$$soundness: \quad x \in A \Longrightarrow g(x) \in B \tag{2.3}$$
$$completeness: \quad g(x) \in B \Longrightarrow x \in A \tag{2.4}$$

2.2.3. NP-Hardness and Satisfiability

To show a decision problem B is **NP-Hard**, we have to find a **NP-Hard** problem and polynomial reduction s.t. $A \leq_p B$. A first proven **NP-Hard** problem is **Satisfiability**, which was independently proven by Cook in 1971 and Levin in 1973. The **Satisfiability** problem is denoted by

Satisfiability
Input: A propositional logical formula in conjunctive normal form.
Output: Is there a valid assignment for this formula?

However, for formal verification, we still need a formal definition using mathematical terms. In the previous implementation of the project, the **Satisfiability** is defined by a list of clauses, with the clauses as the sets of variables.

TODO: Replace this with Isabelle output.

There have been many attempts to solve **Satisfiability** problem in a polynomial bound, as well as many approaches to solve **Satisfiability** problem efficiently in certain scenarios. Thus, **Satisfiability** is one of the most studies **NP-Hard** problems, from which there are also many **NP-Hard** problems reduced. Our first reduction also stems from **Satisfiability**, while all the other reductions are constructed upon novel introduced problems. More details on the reduction and implementation are given in Chapter 3 and Chapter 4.

3. Set Covering Problems

In this chapter, we discuss about the **NP-Hardness** of a few set covering problems. Covering problems ask whether a certain combinatorical structure A covers another structure B, or how large A has to be to cover B. We focus on a subclass of covering problems, the exact covering problem. In this subclass, A covers B exactly, i.e. no element in B is covered twice in A. In Karp's paper in 1972, the following covering problems were included: Exact Cover, Exact Hitting Set, 3-Dimensional Matching, Steiner Tree, and Max Cut. In our implementation, we reduced **Satisfiability** to **Exact Cover**, and then reduced **Exact Cover** to **Exact Hitting Set**.

3.1. Exact Cover

We follow the definition given in Karp's paper:

> **Exact Cover**
> **Input**: A set X and a collection S of subsets of X.
> **Output**: Is there a disjoint subset S' of S s.t. each element in X is contained in one of the elements of S'?

3.1.1. Choice of reduction

Since **Exact Cover** is a fundamental **NP-Hard** problem, there are many different reductions available. Karp's reduction is from the Chromatic Number problem. Although the Chromatic Number problem was formalized in Karp's 21 project, we chose not to reduce from this problem because of the complexity of the graph traversal and the slight difference between Karp's definition and the available Isabelle definition. On the other hand, there is an easy reduction from **Satisfiability** to **Exact Cover**. This reduction does not involve graph traversal. The sole implementation hardness is the non-typed set, which is not yet supported by Isabelle. However, this problem is resolvable by lifting the arbitrary data type to a container type.

$$Unfinished \tag{3.1}$$

3.1.2. Details in the reduction

In this part, we discuss the reduction and its correctness in details, with the Isabelle definitions and theorems as a complement.

Reduction Given a propositional logical formula F, we index the variables and the clauses and use the following notations.

1. x_i denotes the i-th variable in the formula with $x_i \in vars\ F$
2. c_i denotes the i-th clause in the formula with $c_i \in F$
3. p_{ij} denotes the j-th position/literal in the i-th clause with $p_{ij} \in c_i$

Then we construct a set X and which contains all 3 different kinds objects.

$$X = vars\ F \cup F \cup \bigcup_{c_i \in F} c_i$$

Furthermore, we construct S, a collection of subsets of X. We determine the following subsets

1. $\{p_{ij}\}$. The unary set of positions
2. $\{c_i, p_{ij}\}$. The binary set of a clause and a position in it.
3. $pos(x) := \{x_i\} \cup \{p_{ab} | p_{ab} = x_i\}$. The set of its positive occurrences as positions.
4. $neg(x) := \{x_i\} \cup \{p_{ab} | p_{ab} = \neg x_i\}$. The set of a variable with its negative occurrences as positions.

. S contains all of the four types of subsets.

$$\begin{aligned} S = &\{p_{ij} | p_{ij} \in c_i, c_i \in F\} \cup \{\{c_i, p_{ij}\} | p_{ij} \in c_i, c_i \in F\} \\ &\cup \{\{x_i\} \cup \{p_{ij} | p_{ij} \in c_i, c_i \in F\} | x_i \in vars\ F, x_i = p_{ij}\} \\ &\cup \{\{x_i\} \cup \{p_{ij} | p_{ij} \in c_i, c_i \in F\} | x_i \in vars\ F, \neg x_i = p_{ij}\} \end{aligned}$$

The pair of (X, S) is the input for the **Exact Cover** problem.

Soundness Let F be a satisfiable propositional formula, with $\sigma \models F$ as a valid assignment. We construct an exact cover $S' \subseteq S$ of X in the following steps.

1. For each $x_i \in vars\ F$, $pos(x)$ is included in S' if $\sigma(pos(x)) \equiv \top$. Otherwise we insert $neg(x)$ into S'.
2. For each $c_i \in F$, we choose the minimal j with $\sigma(p_{ij}) \equiv \top$, and insert $\{c_i, p_{ij}\}$ into S'
3. For each $p_{ij} \in c_i$, if $\sigma(p_{ij}) \equiv \top$ and $\{c_i, p_{ij}\}$ is not in S', the unary set $\{p_{ij}\}$ is included.

Obviously, each position in included $pos(x)$ and $neg(x)$ will have the false value under the assignment σ, while the positions in the other sets all have truth value. By design, the positions in the second and the third steps never duplicate. Thus, the positions never occur in two different sets in the collection S'. Furthermore, the clauses and variables occurs in exactly one set in S'. Hence the constructed collection is disjoint.

From the sole occurrence of the clauses and variables, we can also conclude that they are covered in this collection. Now we only have to prove that all the positions are covered. If a position p_{ij} has the false value under σ, it is covered in the first step. Otherwise it is either covered in the second step or the third step. With the disjointness, we may conclude that S' covers X exactly and that (X, S) is an instance of the **Exact Cover**.

Completeness Let (X, S) be an exact cover pair which is reduced from the propositional logical formula F. It is easy to reconstruct the model σ with the same approach as in the proof of the **soundness**, which also shows that F is satisfiable.

A. General Addenda

If there are several additions you want to add, but they do not fit into the thesis itself, they belong here.

A.1. Detailed Addition

Even sections are possible, but usually only used for several elements in, e.g. tables, images, etc.

B. Figures

B.1. Example 1

✓

B.2. Example 2

✗

List of Figures

List of Tables