NExt ApplicationS of Quantum Computing

D6.6: Divide and quantum open source software

Document Properties

Contract Number	951821
Contractual Deadline	M18
Dissemination Level	Public
Nature	Other
Edited by :	Sebastiaan Brand, Leiden University
Authors	Sebastiaan Brand, Leiden University, Alfons Laarman, Leiden University
Reviewers	Vicente Moret, Universidade da Coruña, Andrés Gómez, CESGA
Date	21/02/2022
Keywords	Quantum Algorithms, Fault Trees, Satisfiability, Grover
Status	Final
Release	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951821

History of Changes

Release	Date	Author, Organization	Description of Changes
0.1	22/02/2022	S. Brand, Leiden University	Initial document
0.2	23/02/2022	S.Brand, Leiden University	Processed comments Vicente and Andrés
1.0	24/02/2022	S.Brand, Leiden University	Changed version number to 1.0

Table of Contents

1. Executive Summary

Fault trees are a type of model which captures how small failures in probabilistic systems can propagate and ultimately lead to a critical system failure. An important component of fault tree analysis is finding small subsets of events which can cause a critical failure (often called “cut sets”). Finding these small cut sets is important because they often correspond to the most likely way a system will fail.

In this deliverable we provide an open source implementation of a procedure which computes minimal cut sets from fault trees by translating the problem to a satisfiability (SAT) problem. This SAT formula can then either be solved with a classical SAT solver, or with a quantum algorithm: specifically Grover’s algorithm for amplitude amplification. The solutions found by both methods are the same, but the quantum algorithm allows for a quadradic speedup in time complexity. Additionally, for quantum computers which have too few qubits to handle the entire problem instance, a divide and conquer approach can theoretically be used to split the problem up and obtain smaller quantum speedups (Rennela, Brand, Laarman, & Dunjko, 2021).

The main component of this deliverable is the open source library itself, which can be found online here: https://github.com/NEASQC/ft-2-quantum-sat. In section 2 of this document we give an overview of problem of finding cut sets, and how the library solves this problem.

2. Finding Cut Sets in Fault Trees with SAT and Grover

This section contains a brief overview of the background to the problem and our implementation.

2.1. Background

Fault trees are a commonly used way to model critical failures in probabilistic systems (Ruijters & Stoelinga, 2015). The simplest type of fault tree is a directed, acyclic graph (DAG) with three types of nodes: leaf nodes which represent “basic events”, internal nodes given by Boolean logic gates, and a single root node representing the “top event” (usually some critical failure). One of the types of information we would like to extract from these fault trees is so called minimal cut sets. These cut sets are subsets of basic events which trigger the top event in the fault tree. A cut set is *minimal* when no subset of elements can be removed without invalidating it.

An example is given in Figure 1. This figure shows a fault tree with 3 basic events (A, B, and C), two internal nodes given by AND gates, and a single root node given by an OR gate. The cut sets of this fault tree are {A, B, C}, {A, B}, and {B, C}, where {A, B, C} is not a *minimal* cut set since elements from this cut set can be removed while still retaining a valid cut set.

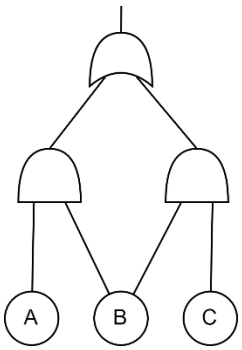

Figure 1: Example of a small fault tree.

Since a fault tree is effectively a logic circuit, the problem of finding minimal cut sets can naturally be translated to a satisfiability (SAT) problem. We assume that the reader is familiar with satisfiability and the use of SAT solvers. For a comprehensive overview of SAT and SAT solving algorithms we refer to for example (Biere, Heule, & van Maaren, 2009). Finding cut sets with classical SAT solving algorithms has been done previously (Barrère & Hankin, 2020). The process of solving SAT problems can be sped up quadratically using quantum algorithms, specifically Grover. Additionally, these SAT problems also lend themselves to be split up into smaller subproblems, to achieve smaller speedups on quantum computers with a limited number of qubits (Rennela, Brand, Laarman, & Dunjko, 2021).

2.2. Overview of Method

The goal of the library is to compute minimal cut sets from fault trees. These fault trees can either be manually constructed with the provided library functions, or be loaded from an XML file in the Open-PSA Model Exchange Format (Steven & Antoine, 2007). Computing minimal cut sets is done by translating the cut set problem to a Boolean satisfiability (SAT) problem: specifically, a Boolean formula in conjunctive normal form (CNF) is created from the fault tree. In order to find the smallest cut sets first, the CNF formulas are additionally constraint with a cardinality constraint over the basic events of the fault tree (Bailleux & Boufkhad, 2003). The library provides the option to solve these CNF formulas using either Grover or a classical SAT solver (specifically we use Glucose (Audemard & Simon, 2018) as the classical solver). One of the benefits of having the option to use a classical SAT solver is that the entire procedure can be tested independently of the quantum part.

Figure 2 gives a high-level overview of the whole algorithm. Note that by iterating over the cardinality constraint **k** starting from 1, and blocking previously found cut sets, we are guaranteed that all the cut sets we find are, aside from small, also *minimal* cut sets.

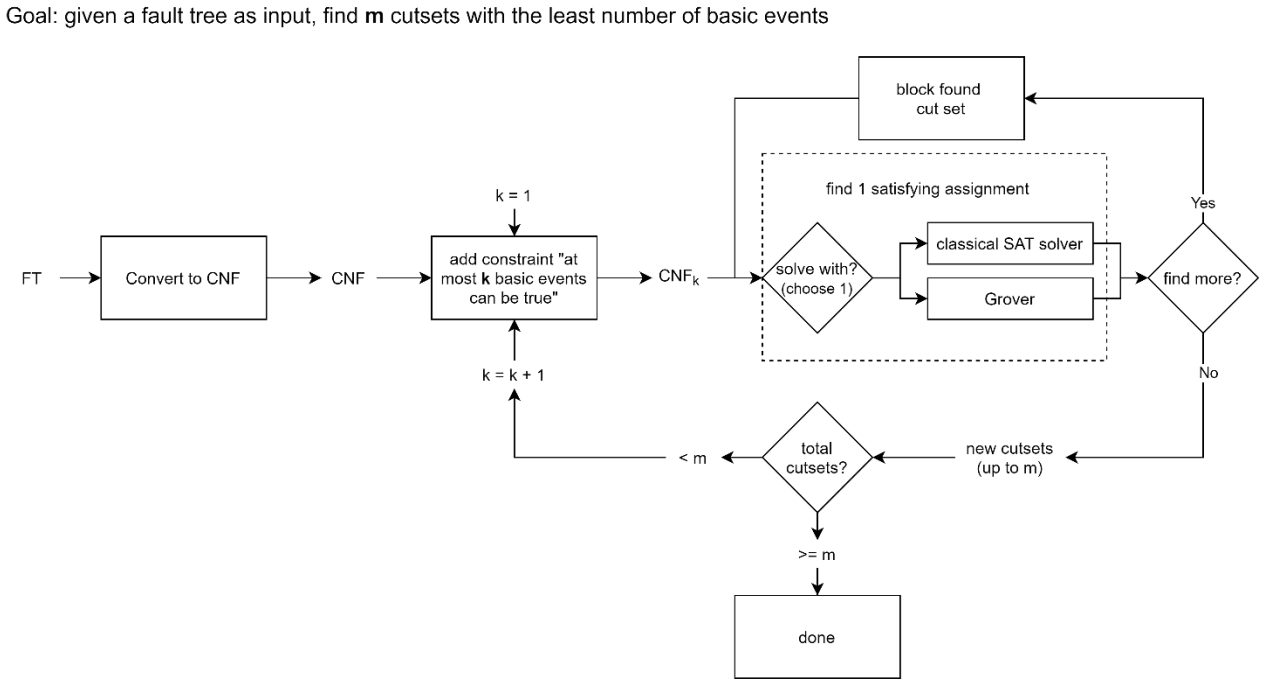

Figure 2: Overview of finding minimal cut sets in fault trees with SAT and Grover.

2.3. Availability and Documentation

- The code is publicly available on the NEASQC Github and can be found here: https://github.com/NEASQC/ft-2-quantum-sat.
- The complete documentation can be found here: https://neasqc.github.io/ft-2-quantum-sat/ and is also referenced on the main repository page.
- The README on the main repository page contains complete instructions on how to use the code.

3. Acronyms and Abbreviations

Term	Definition
CNF	Conjunctive normal form
DAG	Directed acyclic graph
FT	Fault tree
SAT	Boolean satisfiability problem

Table 1: Acronyms and Abbreviations

4. List of Figures

5. List of Tables

6. Bibliography

Audemard, G., & Simon, L. (2018). On the glucose SAT solver. *International Journal on Artificial Intelligence Tools*, 1840001.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF encoding of boolean cardinality constraints. *International conference on principles and practice of constraint programming* (pp. 108-122). Springer.

Barrère, M., & Hankin, C. (2020). Fault tree analysis: Identifying maximum probability minimal cut sets with MaxSAT. *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)* (pp. 53-54). IEEE.

Biere, A., Heule, M., & van Maaren, H. (2009). *Handbook of satisfiability.* IOS press.

Rennela, M., Brand, S., Laarman, A., & Dunjko, V. (2021). Hybrid divide-and-conquer approach for tree search algorithms. *arXiv preprint arXiv:2007.07040*.

Ruijters, E., & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer science review*, 29-62.

Steven, E., & Antoine, R. (2007). Open-PSA Model Exchange Format. *Rapport technique. The Open-PSA Initiative*.