UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Informatica

Elaborato finale

Improved Techniques for SAT-to-Ising encoding applied to Hybrid Quantum Annealing

Advisor

Roberto Sebastiani

Student

Giuseppe Spallitta

Co-advisor

Marco Roveri

Anno accademico 2019/2020

Ringraziamenti

...thanks to...

Indice

Summary

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of first-order formulas with respect to various theories, such as real and integer arithmetic, bit vector and floating-point arithmetic. In the last decade, multiple SMT solvers have been developed to address these problems and have been applied in real-life applications, mainly in the formal verification field.
Some problem instances require not only to obtain a valid assignment of variables satisfying the formulas, but they also need to retrieve a model which is optimum with respect to a subset of objective functions. For instance, when timed and hybrid systems are taken into consideration, studying the minimum interval of time before an event happens can be useful to ensure safety properties are always valid. In order to deal with these problems, an extension of SMT was proposed known as Optimization Modulo Theories (OMT), adding to the SMT formulation the already cited objective functions to optimize.
While SMT has been extensively studied and its state-of-the-art is quite advanced, OMT is still in its first steps; only a few OMT solvers are available (one of them, OptiMathSAT, is developed in collaboration by UniTN and FBK) and a lot of progress has still to be done. In this thesis we will concentrate on the problem of effectively encoding SAT and MaxSAT problems into Quantum Annealers. The completion of this task is achieved using OMT solvers to determine the mapping of a Boolean Formula into qubits of the annealer, thus showing a real-life application of OMT. In particular we will discuss the problem of quantum co-tunneling, how it negatively affects the encoding provided by the solvers and a proposed approach to contrast this issue.

1 Introduction

1.1 The problem

1.2 The solution

1.3 Structure of the thesis

This thesis is organized as follows: Chapter 2 will offer an overview about Optimization Modulo Theories, Constraint Programming and their relationship. The chapter will also offer an introduction to OMT applications in Quantum computing, focusing on the SAT-to-Ising encoding problem and how Optimization Modulo Theories are applied in this task. Chapter 2 discusses the definition of a novel algorithm to refine the QUBO encoding in order to obtain more stable encodings. Chapter 3 discusses some tests performed to the post-process approach, showing performances and results. Chapter 4 summarizes the most relevant results and gives the conclusions, additionally offering some cues for future research work.

2 Background

This chapter will offer an overview on some relevant concepts essential to understand the core of this thesis. First we will provide the definition of SAT, MaxSAT and a brief overview of the state-of-the-art algorithms used to efficiently solve them. We will also discuss some extensions of the SAT framework, SMT and OMT, citing some popular solvers built in the last decade to deal with these classes of problem. Lastly, an introduction to Quantum Computing and the Adiabatic Quantum Annealing is provided.

2.1 SAT

Computer science problems can be classified according to their computational complexity in returning a solution. Problems which complete their tasks in polynomial time with respect to their input size are known as P problems; on the other hand, NP problems do not provide a sub-exponential algorithm capable of determining the existence of a solution. The class of NP-hard problems relevant to our discussion is known as **Propositional satisfiability problem (SAT)**.
SAT definition is the following: given a formula made up of Boolean variables as input, out task is to determine if each variable of the input formula can be consistently replaced by the values True () or False () so that that the formula evaluates to True. If the condition above can be satisfied, then the Boolean formula is considered **satisfiable**; on the opposite case, they are defined **unsatisfiable**. Let's discuss an example: given the following formula involving 2 Boolean variables (x_1 and x_2):

$$\varphi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \tag{2.1}$$

In this case φ is satisfiable: if we set $x_1 = T$ and $x_2 = F$, the entire formula returns true.
In order for a Boolean formula to be satisfiable, it is sufficient to obtain a valid assignment to solve the problem: this means that multiple solutions could be acceptable for a single formula to prove its satisfiability. The problem complexity grows exponentially with the increase of Boolean variables (when N variables are involved, 2^N possible assignments have to be tested in the worst case to determine its unsatisfiability).
SAT solving found real life applications in HW/SW synthesis and verification problems, cryptography and security issues and many other tasks. Currently SAT solvers are capable of managing up to 10^6 variables and 10^7 disjunctions for some specific case. On the other hand, more complex problems are actually out of reach despite a low number of variables to manage, for instance cryptanalysis and verification of arithmetic circuits. The search of efficiency for these tasks is the ultimate goal of our research.
Boolean formulas can be converted to equivalent formulas with desired properties than can help SAT solvers in determining its satisfiability. The two most popular Boolean conversion are the **conjunctive normal form (CNF)** and the **Tseitin transformation**.
A Boolean formula φ is written in conjunctive normal form is it is structured the conjuction of simpler disjunctive sub-formulas (each disjunction is then called clause), satisfying the following formulation:

$$\bigwedge_{i=1}^{L} \bigvee_{j_i=1}^{K_i} I_{j_i} \tag{2.2}$$

Each Boolean formula can be reduced to an equivalent CNF formulation applying some transformation rules known as **De Morgan's rules**:

$$\neg(\alpha \wedge \beta) = (\neg\alpha \vee \neg\beta) \tag{2.3}$$

$$\neg(\alpha \vee \beta) = (\neg\alpha \wedge \neg\beta) \tag{2.4}$$

The main issue generated by a CNF conversion is its exponential increase in size; on the other hand the second proposed approach, Tseitin transformation is capable of simplify the formula maintaining a linear growth. The key idea is the introduction of auxiliary variables to represent the output of subformulas and then constrain those variables using CNF clauses. In details, given a formula φ we need to follow these steps:

- We introduce a new variable for each subformula ψ of φ.
- We consider each subformula $\psi = \psi_1 \circ \psi_2$, where $\circ$ can be any Boolean connective, and stipulate representative of ψ is equivalent to representative of $\psi_1 \circ \psi_2$.
- Each subformula is converted into its CNF-equivalent formulation.
- We can build the Tseitin formula as:

$$\psi_\varphi \wedge \bigwedge_{\psi_1 \circ \psi_2 \in \text{ subf}(\varphi)} \text{CNF}(\psi_{\psi_1 \circ \psi_2} = \psi_{\psi_1} \circ \psi_{\psi_2}) \tag{2.5}$$

We need to point out how the formula obtained after a Tseitin transformation is not equivalent to its original form (because of the newly introduced variables); despite this the two Boolean formulas are equisatisfiable, meaning that the first formula is satisfiable whenever the second is and vice versa and thus being suitable to be used for our tasks.

2.1.1 How to solve SAT problems: DPLL and CDCL

The first algorithm acquiring popularity to deal with SAT instances is known as **Davis–Putnam Logemann–Loveland (DPLL) algorithm**. It is a complete, backtracking-based search algorithm for deciding the satisfiability of propositional logic formulae in conjunctive normal form. The solver chooses a literal and assign a truth value between True and False; this choice will simplify the formula and the value of a subset of other literals will be bounded. Once no more variable values can be deterministically assigned, we will choose a second variable and repeat the procedure until we retrieve a satisfying assignment or a conflict is found. In the second case, we perform a chronological backtracking: we jump back to the most-recent open branching point and start the search from that point. If no path reach satisfiability, the formula is considered unsatisfiable.
The algorithm described above, albeit working, it quite inefficient because of the amount of resources wasted to perform each backtrack jump. To solve this issue, a variant of the algorithm has been proposed and it is the current state-of-the-art approach implemented in the SAT solvers: **conflict-driven clause learning (CDCL)**. The algorithm behaviour is similar to DPLL until we reach a conflict: in that case information is collected to efficiently backtrack and avoid lots of redundant search. In particular CDCL relies on:

- **Conflict analysis**: once a branch fails, we determine the subassignment η causing the conflict, determining the conflict clause.

- **Learning**: the conflict clause is stored by the solver into the conflict set, ready to be applied when required.
- **Backjumping**: we can then backtrack to the highest branching point so that the stack contains all-but-one literals in η, and then unit-propagate the unassigned literal on C.

While this approach drastically prunes the search space, we need to take into account the growth of space complexity required to store clauses. Heuristics have been proposed to determine when a learned clause is no more useful for the task in question and can be dropped.

2.1.2 Representing SAT using And-Inverter Graph

Each Boolean formula can be represented using different approaches. The structural implementation relevant to the scope of this dissertation is known as **And-Inverter Graph (AIG)**. An AIG is a directed, acyclic graph characterized by:

- Terminal nodes, representing Boolean variables.
- Two-input nodes, representing logical conjunctions
- Marked edges, representing negation of a logical conjunction.

To convert a Boolean problem into an equivalent AIG representation, it is necessary to a series of 2-input AND gates and negations (the application of CNF and De Morgan's Laws helps us in achieving the result).

2.2 MaxSAT

A subclass of SAT problems is called weighted maximum satisfiability, or MaxSAT. A MaxSAT formula can be seen as optimization extension of SAT: the formula is defined as a conjunction of clauses and, for each clause, a weight (a positive real number) is assigned. Given a Boolean assignment of its variables, a score is calculated considering the weights of all satisfied clauses. This addition change the perspective of the problem: every assignment can now be considered acceptable, since some clauses can be falsified. As a consequence MaxSat defines a new task: find a Boolean assignment so that the associated score is maximum. Also in this case multiple assignment can obtain the maximum score, resulting in multiple valid assignments.
To better understand it, we will cover it using an example.We will consider the following Boolean formula:

$$\varphi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \tag{2.6}$$

Each clause has also an additional weight: first clause from left has a score of 1, each subsequent clause's weight is higher by 1 with respect to the precedent. Clearly no assignment could satisfy all clauses at the same time, but this has no interest since we are solving a MaxSAT problem. Quickly considering all possible assignments, we can state that the one guaranteeing the maximum total score is $x_1 = F$ and $x_2 = F$, satisfying respectively the clauses with weights 2, 3 and 4. No other Boolean assignment could achieve the same result, so this is the only solution to our MaxSAT problem.

2.3 Satisfiability Modulo Theories

Satisfiability Modulo Theories represents an extension to the SAT problem. The input and the final goal remain the same as SAT: this means that we have to deal with first-order logic formulas and we search for a valid assignment of variables satisfying such formulas. The main

difference relies on the fact that binary boolean variables can be replaced by predicates and functions belonging to non-binary theories. Some of the most relevant theories in the Formal Verification fields are:

- Linear Arithmetic over Integers, Real or both (LIRA)
- Bit vector arithmetic (BV)
- Floating-point arithmetic(FP)
- Uninterpreted functions (UF)

Given the non-binary nature of these theories, novel approaches have been studied and tested to efficiently evaluate the satisfiability of a given formula. At the current state, the most widely used algorithm is known as Lazy SMT solving, whose core idea will be briefly explained to the reader for a better comprehension. Given an SMT formula and a subset of theories, we produce a Boolean abstraction formula in which each non-binary predicate is replaced by a binary variable. This new formula is then fed to a CDCL SAT solver and used to search a satisfiable assignment. When a satisfiable assignment is obtained, the corresponding set of T-literals that make up the original problem are fed to specialized theory solvers. If each clause if T-consistent, then we proved the satisfibility of the original formula; otherwise, the conflict clauses causing unsatisfiability are extracted and learned by the CDCL algorithm , permitting the evaluation of new assignments. The algorithm goes on until satisfiability is proved or no more assignment can be returned, proving its unsatisfiability.
In order to be evaluated, the instance of a SMT problem has to be instantiated using a standard format. Regarding SMT encoding, SMT-LIB is the international initiative on top of which most SMT solvers are built. SMT-LIB does not only promote common input and output languages for SMT solvers; it also provides standard rigorous descriptions of background theories used in SMT systems and establishes benchmarks that can be used to test the solvers to highlight weaknesses and strengths.
The syntax of SMT-LIB-based tools is quite simple. First,we define properties of the problem and solver options (for instance determine if we want to extract unsat cores). We can also set a theory, so that efficient algorithms and heuristics are applied during execution to improve performances. After this introduction, we declare constants, variables and functions. Then assertions are expressed: they bind the values of variables, restricting the range of satisfying solutions. A simple example is provided in listing 1.1.
Among the solvers accepting SMT-LIB as input format, University of Trento and the Bruno Kessler Foundation have developed a SMT solver, called MathSAT. The code, written in C++, support all the theories mentioned above, plus other logics as Arrays arithmetic. It also supports interesting functionalities such as generation of models and proofs for satisfiable problem, extraction of unsatisfiable cores for the unsatisfiable ones and incrementality.

2.4 Optimization Modulo Theories

Optimization Modulo Theories represents an extension to the SMT framework. In addition to the already described first-order formulas involving non-binary formulas, we focus on defining one or more non-binary objective function. As a result, retrieving a satisfying assignment is not sufficient anymore: the goal is now obtaining a valid model while minimizing/maximizing the objective functions, according to our choice.
The language used to encode OMT problems is an extension of the SMT-LIB standards. In addition to the syntax already discussed for SMT-LIB, some commands are available to define

```
(set-info :smt-lib-version 2.6)
(set-option :print-success false)
(set-option :produce-models true)

(declare-const x Int)
(declare-const y Int)
(declare-fun f (Int) Int)

(assert (= (f x) (f y)))
(assert (not (= x y)))

(check-sat)
(get-value (x y))
```

Listing 2.1: An example of SMT-LIB encoding.

the cost functions we desire to optimize and the direction of optimization. The structure of this command is:

solve [maximize/minimize] <cost function>

If permitted by the solver, we can also write multi-objective optimization problems. multi-objective optimization problems require the presence of some instructions to determine the philosophy adopted to determine the goodness of a solution. Example of valid approaches are:

- **Pareto Optimality**: given two different solutions and a set of n cost functions $x_1...x_n$, we state that the first solution is better to the second one according to this criterion if there exists a cost function x_i (where $1 \leq i \leq n$) so that x_i calculated on the values of the first solution dominates the value obtained using the second solution; in addition to that, for each cost function x_i the first solution obtain better or equal values than the second one. When using this criterion, multiple solutions can satisfy these conditions: the set of these valid assignment is called **Pareto front**.
- **Lexicographic Optimality**: first we define a total order among the various cost functions, determining a hierarchy. Given two different solutions, we will first compare the value of the first cost function on this hierarchy for both of them; if one of these solutions dominates the other it is chosen as, otherwise we will compare the value associated to the second cost function and so on until reaching the last one.

The jointly work between FBK and University of Trento gave birth to an Optimization Modulo Theories solver, *OptiMathSAT* [5]*.

2.5 Quantum Annealer

In order to achieve efficiency in solving the hardest tasks, one of the proposed techniques involves the exploit of quantum technologies. In the case at issue Quantum Annealers (QA) have been proved to be effective for this task.
Quantum annealers are specialized chips that exploit particular quantum effects (such as superposition and tunneling) to sample or minimize energy configurations. Each configuration is composed by binary variables z_i which can assume a value -1 or 1, representing their state. The energy configuration can be represented using the Ising Hamiltonian function, whose structure is the following:

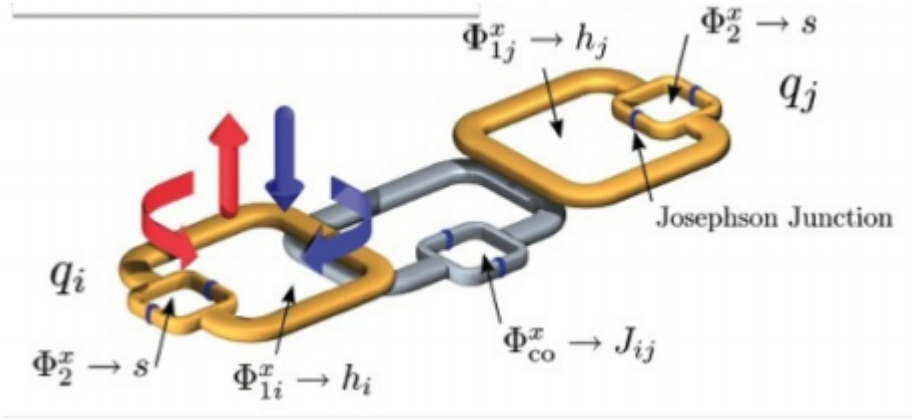

Figura 2.1: A physical representation of qubits in an Quantum Annealer.

$$H(\underline{\mathbf{z}}) = \sum_{i \in V} h_i z_i + \sum_{\langle i,j \rangle \in E} J_{ij} z_i z_j \tag{2.7}$$

Formula 1.3 shows the parameters that influence the system's behaviour:

- h_i are called **biases** and are defined as a real number in range $[-2, 2]$
- J_{ij} and lastly J_{ij} are called **couplings** to the connected pair of qubits at index i and j and is a real number in range $[-1, 1]$.

The search of the ground state of an Ising model is known as **quadratic unconstrained binary optimization (QUBO)** problem.

2.5.1 The SQUID transistor

The minimal unit composing a quantum annealear is called **qubit**: a qubit can assume a symbol value of 0, 1 or a superimposition of 0 and 1. The physical devices used to build qubits are the **Superconducting QUantum Interference Devices (SQUID)**, where the word interference refers to the electrons patterns that give birth to quantum effects. The structure of this qubit transistor is defined by two superconductor coupled by a weak link, usually a thin insulating barrier. Transistors satisfying these conditions are subject to the **Josephson effect**. The superconducting qubit structure instead encodes 2 states as tiny magnetic fields, which either point up or down. We call these states +1 and -1, and they correspond to the two states that the qubit can 'choose' between. Using the quantum mechanics that is accessible with these structures, we can control this object so that we can put the qubit into a superposition of these two states as described earlier. So by adjusting a control knob on the quantum computer, you can put all the qubits into a superposition state where it hasn't yet decided which of those +1, -1 states to be. [Modificare]
Each ring can be superimposed to other rings, defining more complex architectures and admitting an exchange of information among different qubits (thus the definition of couplings in the Ising formula). Figure 1.1 shows the physical structure of a qubit.

2.5.2 D-Wave Quantum Annealers

The entire energy configuration can be represented as a connected graph, where vertexes represent our qubits and edges represent connections between qubits. This configuration highlights

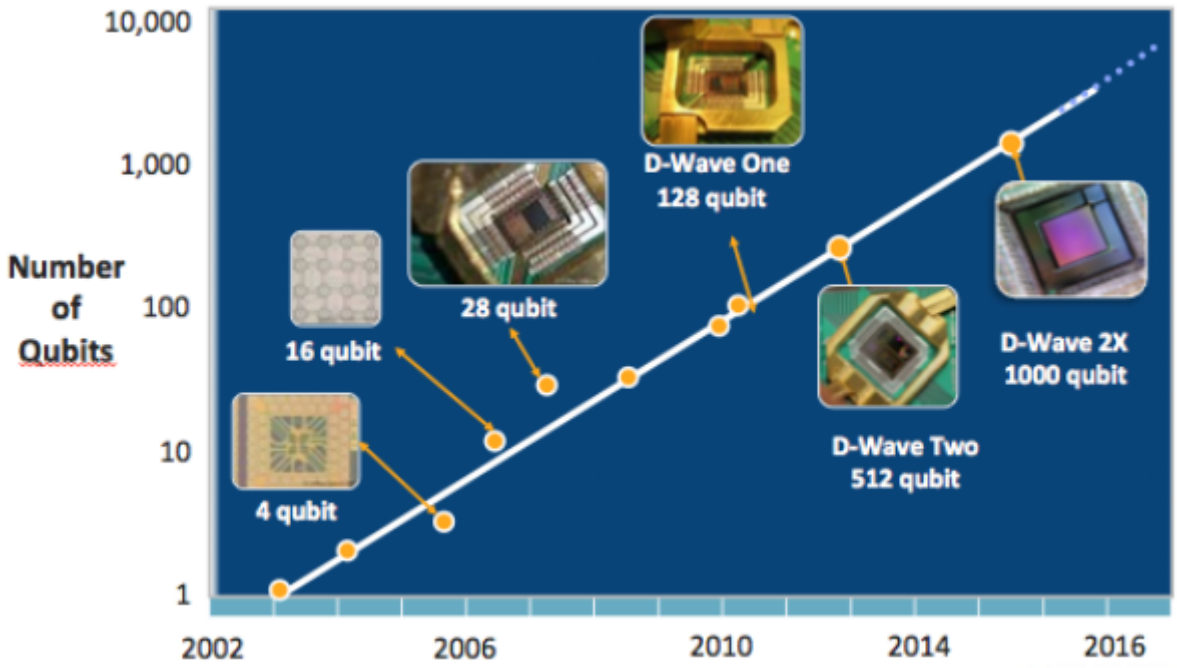

Figura 2.2: D-Wave annealers growth over the years. Courtesy of D-Wave Systems Inc.

how qubits' ground state is not considered independently, but their interaction with neighbour qubits alters the final energy state of our configuration. The choice of a specific quantum annealer influences the formulation of an Ising model: each annealer has a specific architecture and specific properties, constraining the formulation of a problem. In this thesis we will concentrate on annealers developed by the Canadian company D-Wave [2]. In the beginning, the Canadian company aimed in increasing the number of available qubits, obtaining Moore-like growth as shown in figure 1.2. In a second phase, the company put its effort in defining novel structures with better connections and fewer wasted qubits. Currently D-Wave has already developed and produced a first quantum system, known as **Chimera**. The basic features of this architecture are:

- The basic unit is the tile, which contains 8 qubits.
- Qubits of a cell unit are grouped into two groups (the horizontal and vertical qubits): qubits belonging to the same group are connected to qubits of the opposite group thanks to couplings.
- Unit cells are tiled vertically and horizontally with adjacent qubits connected, creating a 16×16 lattice of qubits. The number of total qubits
- From previous constraints, we can see how each qubits can be connected to a maximum of 6 other variables, resulting in the sparsity of the matrix. Qubits that are not connected to each other are managed while writing the Ising formulation setting the coupling value as 0.

In addition to this structure, D-Wave is already studying new architectures overcoming the sparsity of the graph and the absence of cliques; in 2019 the company proposed a new architecture called **Pegasus**. The properties of this system are:

- The number of qubits is $24 * N * (N - 1)$, where N is an integer number.

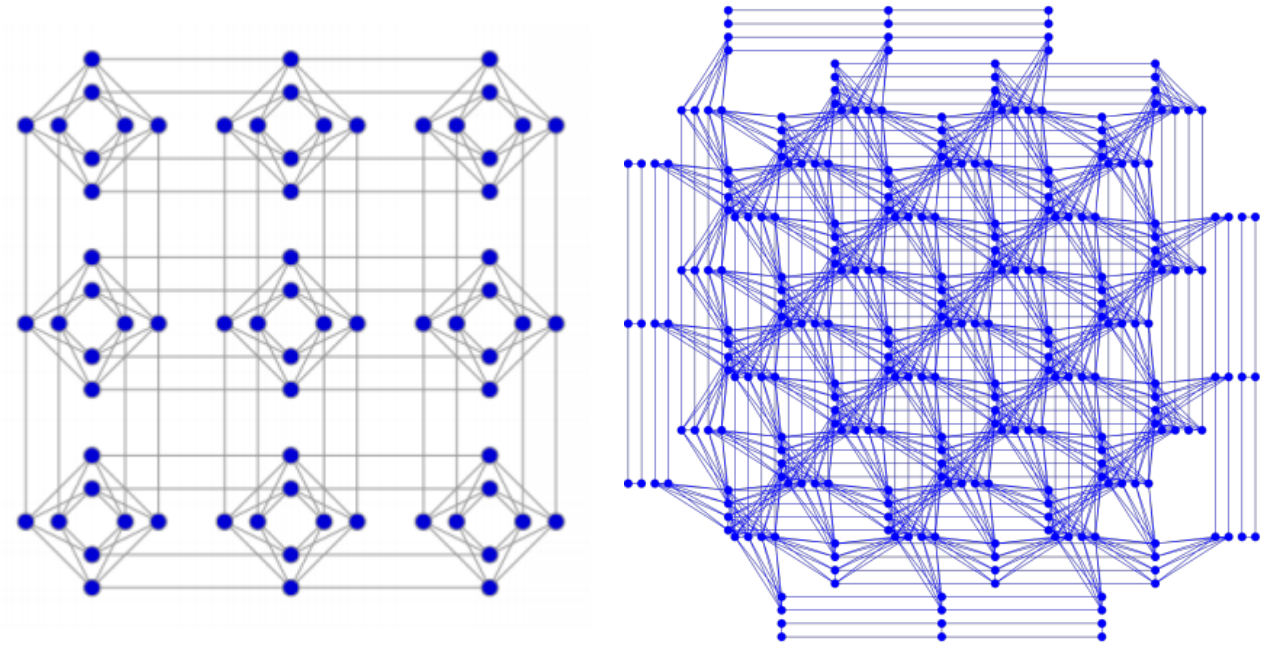

Figura 2.3: A representation of the two architecture proposed by D-Wave, a $3*3$ tiles sub-graph of Chimera (on the left) and Pegasus (on the right).

- The architecture is less modular than Chimera and, in particular, it is not structured in 8-qubit tiles.
- Pegasus graph is less sparse than Chimera, presenting an higher number of interleavings among qubits (exactly 15 couplings for a single qubit).
- The increasing number of couplings have given the opportunity to obtain huge progress with respect to the old model. In particular, Pegasus provides 3 and 4-cliques and qubit duplication.

Figure 1.3 shows the graphs representing both Chimera and Pegasus topologies, rapidly displaying the main differences between the two models.

3 Related work

In this chapter will deepen on the SAT-to-Ising reduction problem, which has been proved to be useful when trying to exploit properties of quantum computing to reduce the computation time to solve SAT problems. We will discuss a preliminary algorithm that have been studied at University of Trento, distinguishing the procedure adopted to encode simple formulas from complex Boolean instances.

3.1 SAT-to-Ising

As already stated in the previous paragraphs, SAT main issue is the exponential growth of complexity, stopping computer scientists in investigating hard tasks involving a small number of variables. Quantum computing, on the other hand, exploit specific phenomena such as tunneling to search the minimum of an Ising problem, optimizing computational time. If we would be able to encode SAT/MaxSAT problem into an Ising Hamiltonian problem, we could exploit quantum calculus properties to verify satisfiability of out original problems, drastically reducing computational time. The task described is known as **SAT-to-QUBO**.
Computer scientists use Quantum Annealers as black-box algorithms which are fed to QUBO problems, whose formulation is similar but not identical to an Ising problem:

$$H(\underline{\mathbf{z}}) = \theta + \sum_{i \in V} h_i z_i + \sum_{\langle i,j \rangle \in E} J_{ij} z_i z_j \tag{3.1}$$

As we can see qubits, biases and couplings are present in our formulation; in addition to them we consider the parameter θ_0, which is called offset and falls into the range $(+\infty, -\infty)$. Given a SAT problem, we are interested in finding a variable placement $\mathbf{x} \to \mathbf{z}$ on the quantum annealer qubits and the values of offset, biases and couplings so that:

$$P_F(\underline{\mathbf{x}}|\underline{\theta}) = \left\{ \begin{array}{ll} = 0, & if\ \underline{\mathbf{x}} \models F \\ \geq g_{min}, & if\ \underline{\mathbf{x}} \models T \end{array} \right\} \tag{3.2}$$

The gap g_{min} is essential to manage sensitivity of the quantum annealer: the optimization will be affected by noise generated by the non-zero Kelvin temperature and , so it will be rarely the case that not satisfying assignment will get 0 as final score. Setting the gap (and trying to find the highest gap possible satisfying the problem) will help us in discriminating acceptable assignments and not acceptable ones. For instance, a formula we can easily encode into an Ising Hamiltonian function is $\varphi = x_1 \iff x_2$. A simple penalty function would be $P_F(\underline{\mathbf{x}}|\underline{\theta}) = 1 - x_1 x_2$, whose gap for satisfying assignment is 2.
The current formulation of our problem presents some issues:

- Chimera tile architecture is quite limited: the absence of clique and the sparsity of the matrix reduce the number of problem we can represent.
- The problem is actually overconstrained: you need to search your solution checking $O(2^{|\underline{\mathbf{x}}|})$ models/countermodels. You have also to deal with the degrees of freedom of its formulation caused by biases and couplings.

To overcome this difficulties, we can add a non-fixed numbers of ancillary variables to provide the missing links. The problem will slightly change into the task:

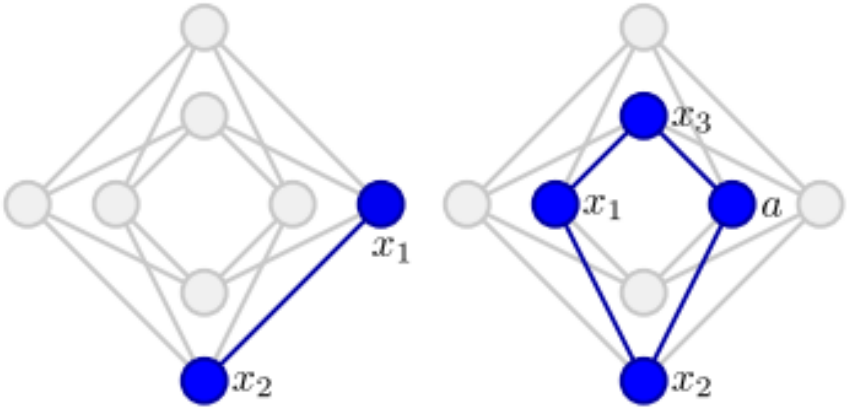

Figura 3.1: On the left, a valid positioning of qubits into a Chimera tile for the formula $\varphi = x_1 \iff x_2$. On the right, a valid positioning for the formula $\varphi' = x_3 \iff (x_1 \vee x_2)$. Both architectures are inspired by the penalty functions discussed in this paragraph.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \left\{ \begin{array}{ll} = 0, & if\ \underline{\mathbf{x}} \models F \\ \geq g_{min}, & if\ \underline{\mathbf{x}} \models T \end{array} \right\} \tag{3.3}$$

The more ancillary variables we add to our formulation, the more complex the resulting task will be, so we will always try to limit their use only when necessary. Ancillary variable are essential for basic encoding formulas: an example could be $\varphi = x_3 \iff (x_1 \vee x_2)$. If the Quantum Annealer admitted cliques the encoding would be trivial; since this is not the case, an ancilla is added to obtain a valid penalty function:

$$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a \tag{3.4}$$

Figure 1.3 shows the positioning of the obtained Ising Hamiltonian functions into a Chimera tile.

3.1.1 Determining the penalty function

The task of retrieving the parameters characterizing the penalty function of a Boolean formula can be espressed by a SMT problem:

$$\forall \underline{\mathbf{x}} \left[\begin{array}{l} (F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) \geq 0)) \wedge \\ (F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = 0)) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) \geq g_m in)) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = g_m in)) \wedge \end{array} \right] \tag{3.5}$$

The last row of equation (2.2) can be omitted if we are not searching an exact penalty function. Since this omission drastically reduce the computational time to search a valid solution, we will usually consider it omitted.

Classic satisfiability theories do not deal with quantifiers, so it is essential to successfully remove them, obtaining and equivalent formula. The most popular approach adopted is **Shannon expansion**. Given an universal quantifier with respect to a set of variables, we can build an equivalent formula as the union of set of clauses whose structure is identical to the original structure, additionally setting the variables inside the considered set to one of their possible value. On the other hand, if we are presented an existential quantifier, we can recycle the previous algorithm but we will connect all the clauses using the AND operator instead of OR.

Clearly the procedure is exponential (the more variables we consider, the more combinations of true/false value we can obtain). Once the expansion takes place, the original problem is reduced to the following SMT problem on linear real algebric theory:

$$\begin{aligned}\Phi(\theta) = & \bigwedge_{Z_i \in \underline{\mathbf{x}},\underline{\mathbf{a}}}(-2 \leq \theta_i) \wedge (\theta_i \leq 2) \\ & \wedge \bigwedge_{Z_i, Z_j \in \underline{\mathbf{x}},\underline{\mathbf{a}}, i<j}(-1 \leq \theta_{ij}) \wedge (\theta_{ij} \leq 1) \\ & \wedge \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}=\top\}} \bigwedge_{\underline{\mathbf{a}} \in \{-1,1\}^h} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) \geq 0 \\ & \wedge \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}=\top\}} \bigvee_{\underline{\mathbf{a}} \in \{-1,1\}^h} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = 0 \\ & \wedge \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}=\bot\}} \bigwedge_{\underline{\mathbf{a}} \in \{-1,1\}^h} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) \geq g_{min} \\ & \wedge \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}=\top\}} \bigvee_{\underline{\mathbf{a}} \in \{-1,1\}^h} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = g_{min}\end{aligned} \tag{3.6}$$

While the SMT formulation will help us in determining if there exists a valid assignment of parameters to apply the SAT-to-Ising conversion, we recall from paragraph 2.1 that determining the highest value that g_{min} can assume is a fundamental task; as a consequence, we can use equation 2.6 to define an OMT problem, where the cost function is simply g_min and the direction of optimization is the search of the maximal.

3.2 Issues in Encoding for Quantum Annealers

Converting a Boolean circuit into a QUBO formulation would apparently seem an easy task to achieve: actually the nature and the architecture of the annealers drastically reduce the number of SAT problems we can embed. The main issues causing the inefficiency are:

- **The number of qubits is not unlimited**: even though the recent architectures provides more than 2000 qubits, they are usually not enough to encode the majority of circuits. This is due by the nature of the two encoding: the number of qubits representing the input width and the circuit size are two different metrics for complexity, where the former is definitively bigger than the latter.

- **The number of couplings is not unlimited**: as already mentioned in paragraph, each architecture can be interconnected to a limited number of other qubits. Consequently, the encoding process has to take into account this upper limit and, if more connections are required, map a Boolean variable into multiple qubits.

- **Noise impacts the system performance**: the ideal condition of a quantum annealer would require a temperature of 0 K and to be heavily shielded by electro-magnetic rays, preserving the properties of the superconductor rings. In reality these conditions are never met and D-Wave system are subject to performance degradation.

Because of the problems described above, advanced algorithms have been studied and tested in order to reduce the amount of qubits required during the reduction of the SAT circuit.

3.3 Encoding complex Boolean formulas

Determine an efficient encoding is decisive, given the limited of number of qubits of Chimera. Penalty functions have some properties we can exploit to iteratively build complex formula from easier ones. The first property is **NPN-Equivalence**: given a boolean formula $F(\underline{\mathbf{x}})$ with its associated penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta})$ and a new Boolean function $F^*(\underline{\mathbf{x}})$ identical to the previous one but a single variable at a generic index i (so that it appears with inverse cardinality in the new formulation), we can recycle $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta})$ to obtain the penalty function for the new problem. In particular, $P_{F^*}(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta^*})$ where θ^* is a new vector of biases and couplings so that for each $z, z' \in \underline{\mathbf{x}}, \underline{\mathbf{a}}$:

$$\theta_z^* = \left\{ \begin{array}{ll} -\theta_z, & if\ z = x_i \\ \theta_z, & otherwise \end{array} \right\} \tag{3.7}$$

$$\theta_{zz'} = \left\{ \begin{array}{ll} -\theta_{zz'}, & if\ z = x_i \vee z' = x_i \\ \theta_{zz'}, & otherwise \end{array} \right\} \tag{3.8}$$

So once we extract the penalty function for a Boolean Formula, its variants can be easily computed switching signs of biases and coupling. For instance, if we had the Boolean formula $\varphi = x_1 \iff -x_2$, we could easily invert signs of couplings and biases associated with x_2, trivially obtaining $P_F(\underline{\mathbf{x}}|\underline{\theta}) = 1 + x_1 x_2$.
The second property fundamental to simplify the task of retrieving penalty functions is the **AND-decomposition**. Given a Boolean function that can be rewritten as a combination of simpler functions so that $F(x) = \wedge_k F_k(\underline{x}^k)$, where each $F_k(x)$ is associated to a penalty function $P_{F_k}(\underline{x}^k, \underline{a}^k|\underline{\theta}^k)$ with minimum gap g^k_{min}, $\underline{x} = \cup_k \underline{x}^k$ and $\underline{a} = \cup_k \underline{a}^k$, then we can define a penalty function for the original formula in the following way:

$$P_F(\underline{x}, \underline{a}|\underline{\theta}) = \sum_k P_{F_k}(\underline{x}^k, \underline{a}^k|\underline{\theta}^k) \ \text{ with } g_{min} = \min_k(g^k_{min})$$

$$\theta_i = \sum_k \theta_i^k \tag{3.9}$$

$$\theta_{ij} = \sum_k \theta_{ij}^k$$

Equation 2.8 works in the case each bias and coupler value satisfies its range. The subformulas making up the decomposition could share some common variables and, in some cases, this could lead to the attainment of out-of-range biases and couplers. This issue can be easily fixed: we can scale up or down the impact of each penalty function adding a weight w_k greater than 0. This addition slightly modifies equation 2.8, determining the general formulation:

$$P_F(\underline{x}, \underline{a}|\underline{\theta}) = \sum_k P_{F_k}(\underline{x}^k, \underline{a}^k|\underline{\theta}^k) * w_k \ \text{ with } g_{min} = \min_k(g^k_{min}) * w_k$$

$$\theta_i = \sum_k \theta_i^k * w_k \tag{3.10}$$

$$\theta_{ij} = \sum_k \theta_{ij}^k * w_k$$

Adding these weights weakens the minimum gap, so it is not used in practice. An alternative to equation 2.9 relies on the renaming of the shared variable, so that each $\underline{x}^k$ is disjoint with respect to the others. To achieve this goal, when two conjuncts F_k, F'_k share a Boolean variable x_i we rename the second occurrence with a fresh variable x'_i and conjoin the two variable using the simple formula $(x_i \iff x'_i)$. Now we can re-define the original formula and its associated penalty function as:

$$F^*(\underline{x}^*) = \bigwedge_k F_k(\underline{x}^{k*}) \wedge \bigwedge_{\{x_i \text{ shared}\}} (x_i \iff x'_i)$$

$$P_{F^*}(\underline{x}^*, \underline{a}|\underline{\theta}) = \sum_k P_{F_k}(\underline{x}^k, \underline{a}^k|\underline{\theta}^k) + \sum_{\{x_i \text{ shared}\}} (1 - x_i x'_i) \tag{3.11}$$

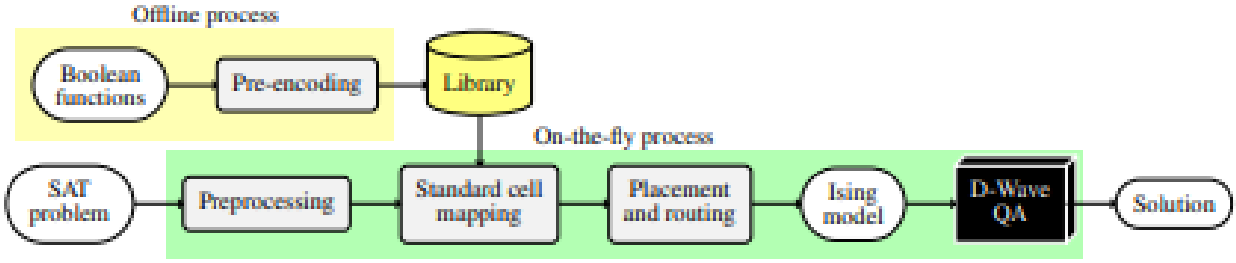

Figura 3.2: A graphical representation of the encoding process for larger Boolean functions.

$$g_{min} = min_k(g^k_{min}, 2)$$

Given the nature of the penalty functions for $(x_i \iff x'_i)$, whose gap is 2, we ensure no issue can emerge and no parameter range error can be observed. Combining the two properties above we can define the procedure to apply to retrieve penalty functions decomposing Boolean formulas into smaller chunks, easier to convert:

- First we Tseitin-style decompose $F(x)$ into an equi-satisfiable formula so that:

$$F * (\underline{x}, \underline{y}) = \bigwedge_{i=1}^{m-1} (y_i \iff F(\underline{x}^i, \underline{y}^i)) \wedge F_m(\underline{x}^m, \underline{y}^m) \tag{3.12}$$

- When two conjuncts F_1 and F_2 share one variable y_j, rename the second with a fresh new variable y'_i, conjoining $(y'_i \iff y_j)$.
- Compute the penalty function for each conjunct separately.
- Sum the penalty functions obtained in the previous step to obtain a single function.

From the algorithm described above, we can define a universal approach to deal with larger Boolean functions, avoiding the application of the SMT formulation which results too much time and resource demanding.
Before starting the encoding, we compute in advance a collection of valid encoding for simple SAT formulas so that they can be mapped using a low number of qubits (in particular when using the Chimera architecture we aim to encode the formula using a single tile). The original SAT formula we want to encode is pre-processed, in order to reduce its size or complexity in terms of its graphical representation. As an example, a good pre-processing approach is using the and-inverted-graph representation, which transform a formula in a combination of AND and NOT functions. Using the simplified structure and the previously computed library we then define a mapping between Boolean variables and the annealer's qubits. This phase represents the most complex and resource-demanding procedure of the algorithm. The fundamental steps of this procedure are:

- Use the library of pre-computed penalty functions to map each function part of the decomposed and simplified SAT formula into a valid QUBO encoding. This phase is known as standard cell mapping.
- Now it is necessary to embed the entire formula onto the QA hardware. To achieve this task, we first assign a disjoint subgraph of the QA hardware graph to each penalty function chosen in the previous step.

- Lastly we need to ensure that qubits representing the same variable are chained so that they can assume the same value when given as input to the annealer: we can accomplish it using penalty functions in the form $1 - x_1 x_2$, where x_1 and x_2 are qubits representing the same variable. This step is a direct consequence to formula 2.8.

The algorithm is heavily discussed in [3] and more details are provided about the choice of and the heuristic adopted to obtain a more stable encoding, but for the sake of brevity we will not discuss it further. Figure 2.4 summarizes the entire process.

4 Tool analysis

In this chapter we will discuss the state-of-the-art tool developed by the jointly work of University of Trento and D-Wave. More specifically we will emphasise critical points of the current version, suggesting possible direction to improve the performance of the algorithm.

4.1 The issue of co-tunnelling

The major issue while converting a SAT problem into a Ising instance is represented by the problem known as **co-tunneling**. Co-tunneling is a side-effect issue caused by the presence of long chains of qubits in the annealer's placement [4]. [Discuss cotunneling in more details]

4.2 Studying the Ising encoding

To determine the behaviour of the placement algorithm and to easily understand how frequently long chains are generated, it was required to graphically represent the annealer architecture when applied to a SAT-to-Ising encoding task.
A new script has been developed to accomplish this task, called *graph_to_dot*. The first step was the building of a graph reflecting the architecture of the quantum annealer chosen at execution. Information about the topology of the network has been retrieved using **D-Wave NetworkX**, an extension of NetworkX providing tools and data for working with the D-Wave systems [1]. Data about the has been collected from the *place_and_route* algorithm: for each AIG-related subformula the algorithm returns a subset of qubits to the graph and the values of weights extracted from the pre-computed libraries. Using this information, we choose a color for each penalty function and use it to determine the related qubits and the involved couplings in the graphical representation. The color of edges representing chains of a Boolean variable have been set to black. An example of Boolean formula placement is shown in figure xx.

The major issue visible from the figure is the presence of long chains of qubits. As already stated in paragraph 3.1, higher size of chains worsen the stability of the annealers in the search of the Hamiltonian final state. Each qubit belonging to the chain can be potentially more useful: unused arcs adjacent to them and set to 0 could be used to determine novel penalty functions. The results would be more complex structures that could help us in obtaining better penalty function, less prone to be subject to co-tunelling and with higher gap_{min}. The script was integrated to the original code as a debug option, permitting other users to test their encoding and better understanding the results behind the implemented procedures.

4.3 The searchPF function

The previous section suggested the idea to recompute penalties function reducing the length of the chains. These nodes, used as ancillas and considering their unused couplings, could provide more complex encoding, reducing the co-tunneling effect and guaranteeing more energy stability. The tool already provide a class of algorithms to compute offset, biases and couplings of an Ising problem, given a topology and the Boolean function. Each function provided some specific constraints in order to satisfying properties such as getting the maximum g_{min} The function relevant to the scope of this thesis, called **searchPF**, was the most suitable to our purpose, given its simplicity and its ease to be extended.
SearchPf requires as input the following objects:

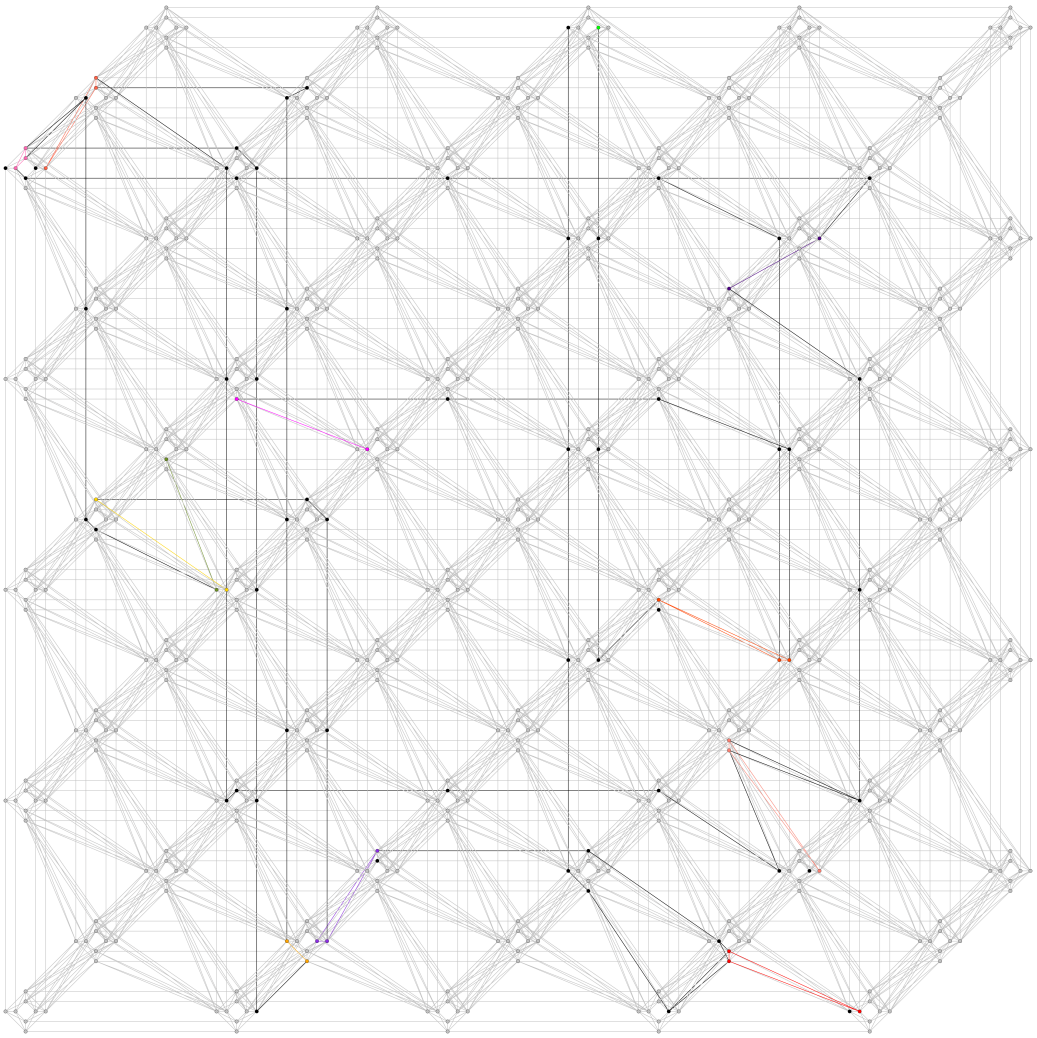

Figura 4.1: A graphical representation of the encoding of a benchmark problem, C17, into the Pegasus architecture.

N. of qubits	Running time
7	1.49 s
8	1.93 s
9	5.01 s
10	33,4 s
11	$>$ 180 s

Tabella 4.1: A table showing the time required to retrieve the penalty function of $A = B \wedge C$ with an increasing number of variables, most of them used as ancillas.

- A graph representing the topology of the involved qubits. Graphs are created using the *GraphViz* library, setting the nodes name as the integer in the range [1,N] with N the number of qubits.
- A function representing the Booolean formula we desire to map into the given graph.
- Two integers nx ans na, representing respectively the number of qubits representing the variables and the number of qubits working as ancillas.

In order to get the weights defining the penalty function, an Optimization Satisfiability problem is instantiated using the previously mentioned parameters. The problem is fed to OptiMathSAT, which returns in response an empty if no valid penalty functions has been found; otherwise, a dictionary containing the parameters value and the minimum gap is returned. Multiple constraints are set to map the problem into the SMT-LIB language:

- **Architecture constraints**: we ensure the resulting penalty function respects the topology of the given graph. As a consequence, some couplings need to be set to 0 a.
- **Range constraints**: As stated in section 1.5, values of biases and couplings need to be constrained so that they cannot violate their ranges. For each coupling and each bias an assert condition setting lower an upper bounds are added to the OMT encoding.
- **Expansion gap constraints**: in order to retrieve the parameters of the Ising function we have to solve the quantified OMT problem described in

The function has been heavily tested to learn its behaviour, its complexity and thus offering cues for improvements. First we studied the relation between the required time to obtain a solution to the OMT problem and the number involved variables, with the goal of determining an upper bound for the number of nodes so that the OMT problems can return their solution in a feasible amount of time to be applied in real-life applications. Table 3.1 shows the results of this analysis; we can see how we are bound to use a very low number of qubits. This is not surprising: solving equation xx requires. More details will be provided in the conclusive chapter, suggesting future approaches to overcome this limit.
We additionally implemented a small section of code to write the output of the procedure on a TXT file, giving us the opportunity to easily understand how *searchPf* generates the problem and its inefficiencies. Some aspect have jumped to our eyes:

- The number of coupling variables generated by the script were higher than the required ones. The architecture constraints tend to create a weight of each pair of qubits, then when the connection is not reflected in the topology of the graph its value is set to 0. While the correctness of the solution is achieved by this approach, an higher number of decision variables lengthens the time to get the solution.

- When an instance is fed to the OMT solver, it tends to return solutions where the connections among ancillas get -1 as value. This result is in contradiction with our goal; most of these parameters should have a value different from -1 not to be part of the chain.

5 Improving SAT-to-QUBO

This chapter will first discuss the weaknesses of the current SAT-to-Ising definition approach, later suggesting a novel algorithm to refine the encoding and retrieve a more stable solution.

5.1 Algorithm

The goal of postprocessing is the re-definition of the QUBO encoding, modifying offset, biases and couplings between qubits so that the longer chains are reduced in size. Since all nodes that make up the chain are linked by couplings whose value is -1, we will ensure to modify these weights.
The only way to achieve this task while not altering the original SAT formulation is to re-compute the scores of each chunk of the simplified formula, defining new OMT problems that involve the additional qubits. In more details, given two penalty functions encoded in a quantum annealer architecture linked by a chain:

- We split the chains into two disjoint sets of nodes. The first set will be associated to the first penalty function, the second to the other.
- We use nodes composing the original penalty functions and the newly assigned qubits of the chain to calculate again the weights of. In particular, we use the nodes belonging to the chain as ancillas for the new function, forcing each coupling between chain nodes and penalty function qubits to be different from zero. We need also to set the most external node of the chain as the actual shared variable, so that we ensure that there is a way to link qubits representing the same Boolean variable and enforcing their equality.
- At this point, we have obtained a subgraph of the original architecture referring to the same. To compute new weights for, we will again rely on formula, but the presence of new ancillas will help us in reducing the number of coupling with score -1. These weights are then used to modify the QUBO encoding before being passed as input to the annealer.

A graphical representation of the algorithm is shown in figure 4.1. Post-processing cannot

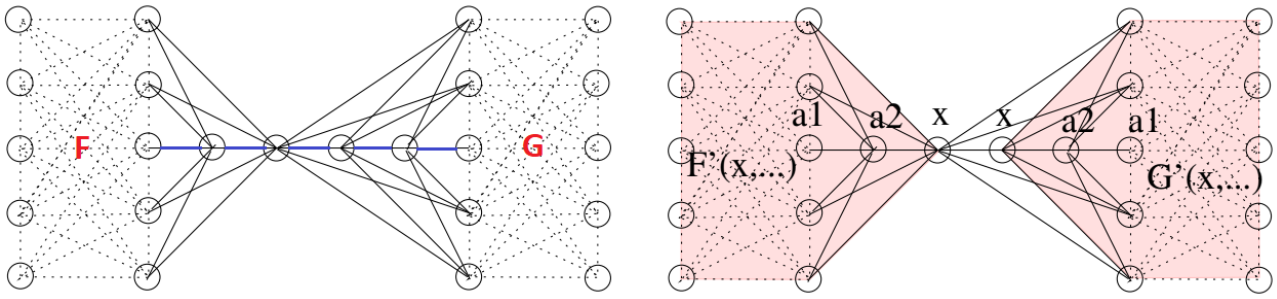

Figura 5.1: A graphical representation of the postprocessing algorithm involving two general penalty functions (F and G) linked by a chain (blue edges). It is important to notice the swap of role between the original node representing the shared variable to ensure consistency of the variable's value between functions.

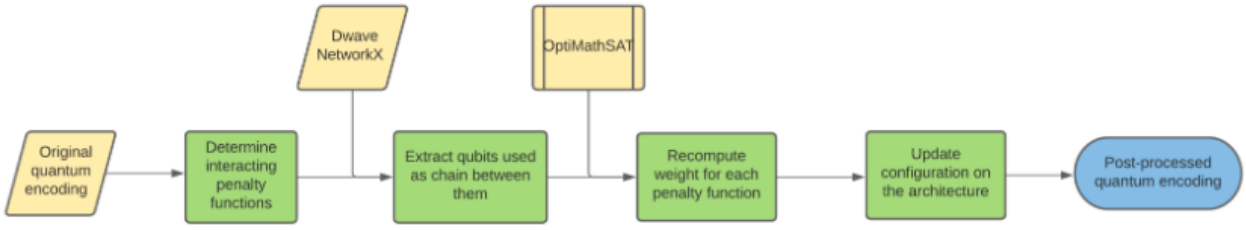

Figura 5.2: A graphical representation of the workflow necessary to implement the postprocessing algorithm. Yellow nodes represent data and state-of-the-art tools; green nodes represent tasks implemented in this thesis.

be applied to every quantum architecture. In order to be successful, it is necessary that qubits presents an high numbers of connections with other qubits. If this condition is not satisfied, we will not obtain great results from the re-computation of scores for penalty functions: nodes belonging to the chain will not link with external qubits and no new couplings to consider will emerge. D-Wave currently proposes two main architectures: the older one, Chimera, is clearly affected by this issue. Actually he graph of its nodes is sparse and a qubits has at most 6 neighbour qubits. On the other hand the most recent architecture, Pegasus, is less sparse and it usually does not fall into the issue described above. As a consequence, from now on we will assume that the algorithm will be executed only on the Pegasus architecture.

5.2 Implementation

While the definition of the approach is linear, in practice we have to deal with numerous issues, mainly caused by computational constraints. As a consequence, the implementation relies on some heuristic that avoid apparent deadlock point during execution. The workflow of the implementation is shown in figure 4.2. More details on each step will be provided in the next paragraphs.

5.2.1 Extracting encoding information

The original version of the code returned the list of values assigned to each parameter, without any additional information about the mapping between Boolean sub-formulas and qubits. While it was not relevant in the precedent version, obtaining this information is now essential to determine how to rearrange qubits in new penalty functions.
To solve the issue, it was required to modify the *place_and_route* library and collect data from the intermediate steps of the procedure. [Continue]

5.2.2 Re-assigning qubits

Once the needed data are stored and available, we can start to post-process the original encoding determining what functions might be extended and what chains are involved. For each pair of penalty functions we decide if there is a chain connecting them which can be split in two disjoint sub-sets of qubits without altering the correctness of the encoding of the original formula.

5.2.3 Recomputing penalty functions

In order to generate the OMT instance necessary to recompute the parameters on the enhanced penalty functions, we need to build the quadruplets of data necessary and call the *searchPF* function using them.
We also tried to enhance the actual encoding, modifying the definition of the SMT file.

5.2.4 Updating the configuration

6 Results

6.1 Ideal model

6.2 Application of CAIG17

7 Conclusions

7.1 Future work

The first direction focuses on the introduction of OMT procedures dealing with non-linearity. At the current state, non-linear constraints, non-linear arithmetics and transcendental functions (such as the logarithm, the exponential and the sine function) are not managed by OMT solvers, in particular by OptiMathSAT. On the other hand, SMT already provides some approaches: in particular, MathSAT has recently been updated introducing them into its decision procedures. As a consequence, it will be useful to extend OptiMathSAT to achieve the optimization of non-linear objectives. The basic idea is to use the optimization routines to generate candidate solutions, then to evaluate them against the non-linear constraints in the problem using some advanced decision procedures.
A second future direction to take into account would focus on bridging the gap between Constraint Programming and Optimization Modulo Theories, the open-source interface described in the previous chapters can be extended to efficiently manage each MiniZinc structure and procedure. To cover the full extent of CP standards some additions are required, such as defining an encoding based on Floating-Point Numbers or dealing with non-linear constraints and objectives (in this case the first future direction will help in completing the task). This may require a general review of the current implementation of the FlatZinc interface, so as to become modular and easy to extend. Moreover, we can improve the situation by developing a T-solver for the theory of sets, which is currently managed using a quite inefficient mix of Boolean and arithmetic constraints. To conclude, another interesting issue concerns CP global constraints, constraints that capture a relation between a non-fixed number of variables. In particular, it would be necessary to study approaches to integrate dedicated procedures to the SMT and OMT framework emulating these constraints.

Bibliografia

[1] D-wave networkx.

[2] D-wave: The quantum computing company. *https://www.dwavesys.com*.

[3] Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, and Stefano Varotti. Solving sat and maxsat with a quantum annealer: Foundations and a preliminary report. In Clare Dixon and Marcelo Finger, editors, *Frontiers of Combining Systems*, pages 153–171, Cham, 2017. Springer International Publishing.

[4] T. Lanting, R. Harris, J. Johansson, M. H. S. Amin, A. J. Berkley, S. Gildert, M. W. Johnson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, E. M. Chapple, C. Enderud, C. Rich, B. Wilson, M. C. Thom, S. Uchaikin, and G. Rose. Cotunneling in pairs of coupled flux qubits. *Phys. Rev. B*, 82:060512, Aug 2010.

[5] Roberto Sebastiani and Patrick Trentin. Optimathsat: A tool for optimization modulo theories. pages 447–454, 07 2015.